Resolving Graphic Conflict In Scale Reduced Maps: A Simulated Annealing Approach

Nathan James Thomas

School of Computing
University of Glamorgan
Prifysgol Morgannwg

A thesis submitted in partial fulfilment of the requirements of the University of Glamorgan for the degree of Doctor of Philosophy.

This research programme was carried out in collaboration with the Ordnance Survey of Great Britain.

May, 2004

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to express my sincere gratitude to my supervisors, particularly Dr. Mark Ware for showing me the error of my ways on more than one occasion. Also thanks to Dr. David Kidner and Professor Andrew Ware, for their support and guidance during this research.

I would also like to thank the members of staff at the Ordnance Survey for their valued assistance and especially deepest thanks to Mr. Simon Gomm.

Finally, a very special thank you to my family and loved ones whose love and support has helped to make this work possible.

Certificate of Research

This is to certify that, except where specific reference is made, the work presented in this thesis is the result of the investigation undertaken by the candidate.

Candidate: .....

Director of Studies: .....

Declaration

This is to certify that neither this thesis nor any part of it has been presented or is being currently submitted in candidature for any other degree other than the degree of Doctor of Philosophy of the University of Glamorgan.

Candidate: ……  ……..

Abstract

Resolving Graphic Conflict In Scale Reduced Maps:
A Simulated Annealing Approach.

Nathan James Thomas

This thesis explores the use of the stochastic optimisation technique of simulated annealing for cartographic map generalisation. The technique performs operations of displacement, deletion, reduction and enlargement of multiple map objects in order to resolve graphic conflict resulting from a reduction in map scale. A trial position approach is adopted in which each of n discrete polygonal objects is assigned k candidate trial positions that represent the original, displaced, reduced and enlarged states of the object. This gives rise to a possible k^n distinct map configurations; the expectation is that some of the configurations will contain reduced levels of conflict. Finding the configuration with least conflict by means of an exhaustive search is, however, not practical for realistic values of n and k. However, the thesis shows through an evaluation of a subset of the configurations, using simulated annealing, can result in an effective resolution of graphic conflict in real time.

Furthermore the thesis explores various methods of improving upon the existing simulated annealing work. Firstly, two techniques were developed that aid in improving execution times using a data partitioning and a two-stage annealing strategy. Secondly, an investigation was carried out which explores the application of high-order feature alignment and the use of a continuous search space. Moreover the thesis explores the use of incorporating additional feature classes into the existing SA framework.

A thorough evaluation has been carried out which demonstrate the usefulness of each approach. The research has achieved five key improvements over the original SA technique: a reduction in execution times; a greater support for generalisation operators; presented a solution to maintaining high-order feature alignment; provided a greater support for additional feature classes; and, a refinement to the search space resulting in an improved graphic display output.

Although the thesis demonstrates the potential of simulated annealing as a means of reducing graphic conflict in scale-reduced maps, there still exists an enormous scope for further work. Additional techniques need to be devised to reduce execution times further for use with on-the-fly generalisation tasks. Other areas for future work include the incorporation of more sophisticated operators and an investigation to determine if SA can be adapted to resolve graphic conflict between linear features.

Chapter 1

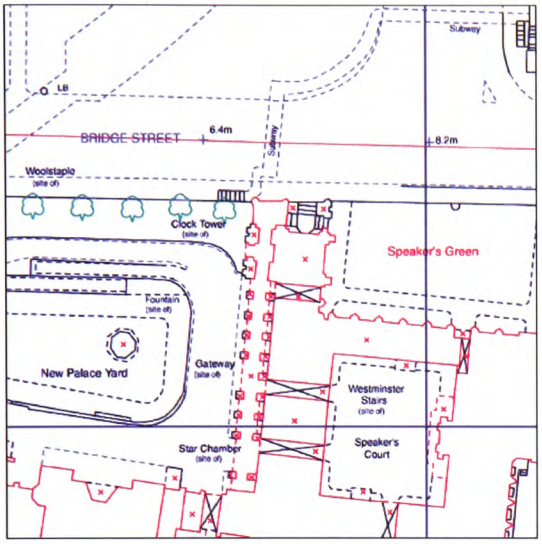
Introduction

This chapter provides background information and describes the motivation for the research. An overview of related concepts and principles associated with digital cartographic generalisation are presented and the aims and objectives of the project are discussed.

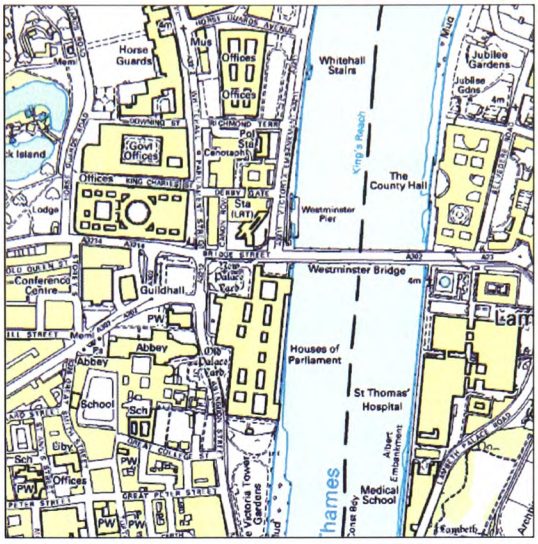
1.1 Digital Cartographic Generalisation

Maps are used to locate places on the surface of the Earth, to show patterns of distribution and to discover relationships between different phenomena by analysing information. When producing a map a cartographer only selects information that is useful to fulfil the purpose of the map. The cartographer presents that information in a symbolised and sometimes modified form suited to the intended scale of the map. The processes of selection, symbolisation and modification are referred to as 'map generalisation'.

Figure 1.1 shows a representation of a series of maps which have been generalised at different scales in order to convey various types of information. These maps are used for a variety of applications. For example: Siteplan® is used in building planning; StreetMap® can be used for tourist activities (e.g. to find important places of interest); while OS™ Travel Map and Landranger® are used for in-car navigation and route-planning.



(a) Siteplan® 1:1,250 scale



(b) StreetMap® 1:10,000 scale

(c) OS™ Landranger® 1: 50,000 scale

(d) OS™ Travel Map 1: 1,000,000 scale

Figure 1.1: Each map represents various types of information at different map scales. Data sets provided courtesy of the Ordnance Survey®; www.ordnancesurvey.co.uk

1.1.1 Definition

Cartographic Generalisation can be defined as the process whereby objects[1] on a map are subjected to a series of specific changes or transformations. These transformations are essential in order to preserve:

i. The characteristics of the map due to a change in scale.
ii. That which is considered to be important, in terms of which symbology is used and how accurate it is represented.

A successfully generalised map must remain legible and produce a clear map image (Rytz *et al.* 1977) and furthermore, preserve essential characteristics of map features (Monmonier 1982). McMaster and Shea (1992) define generalisation as the process of deriving, from a data source, a symbolically or digitally encoded cartographic data set through the application of spatial and attribute transformations.

1.2 Generalisation Transformations

Generalisation transformations are divided into two categories: firstly, conceptual (semantic) generalisation, which concerns changes to the nature of geographical objects; and secondly, geometric generalisation (more commonly known as graphical generalisation), which concerns changes to the physical geometry of an object.

Conceptual generalisation usually takes place prior to graphical generalisation. Its main aim is to simplify a given map through the use of classification and aggregation processes. An example of classification and aggregation is to combine map features, such as a series of streets on a large-scale map, to form an urban area (represented as a shape or outline) on a reduced-scale map.

Geometric generalisations change the geometric characteristics of an object on a map. Examples include the displacement of an object, such as a building, in order to resolve

1. The term 'object' is used to represent a series of points, lines or polygons that represents real-world entities. e.g. buildings, roads, rivers etc.

graphic conflict[2], or the modification of a line so that it contains fewer points. Displacement and point removal are just two of a number of operations that belong to the family of operators that are responsible for the graphical transformations that are applied to objects during the generalisation process. A more detailed explanation of these operators is discussed later, in Section 1.6.

The type of transformation with which this research is concerned is that of graphical generalisation.

1.3 Constraints

Cartographic generalisation is inherently a complicated process. During generalisation map objects are subjected to various forms of transformation, whereby the underlying geometry is modified (to some degree.) It is considered important that these modifications are regulated sufficiently in order to produce a map of acceptable quality. Weibel and Dutton (1998) describe a constraint as, "*a design specification to which the solution should adhere*". There have been a number of constraint-based methods that aim to resolve complex conflict situations. McMaster and Shea (1989) laid down four major principles that should be taken into account when dealing with constraints:

i. Attribute precision.
ii. Geometric precision.
iii. Graphic quality.
iv. Relationship between objects.

Therefore it is important to ensure that:

i. Objects are modified as little as possible.
ii. Objects are recognised in a homogeneous[3] manner.
iii. All objects are clearly distinguishable.
iv. Referential integrity is maintained between objects.

2. 'Graphic conflict' can be described when two or more objects on a map interfere with each other, causing them to be visually as unrecognisable.

3. The term 'homogeneous' refers to the uniform nature of objects that are similar in some aspect.

1.4 Manual Cartographic Generalisation

A large proportion of the map generalisation process carried out at mapping agencies such as the OS™ and the Institut Géographique National® (IGN™) is performed manually by skilled cartographers. An example of an early traditional generalisation process involved the following procedure: An area to be shown in a new, reduced-scale map (the target map) is covered by a set of larger maps or plans (which are gathered together to make the source map). Map features are then chosen from the source map and generalised in a variety of ways before being placed on the target map (Bundy, 1996).

More modern cartographic methods now make use of desktop workstations and Personal Computers (PC's), together with commercial geographical information systems (GIS) applications such as ESRI's™ ARCMAP™ software in an attempt to advance upon previous methods. A cartographer now has the capability of selecting map objects on-screen and then performing the necessary generalisation (this is still a manual process but using a computer aided design-like environment). This has been made possible by new digital mapping products, such as OS™ MasterMap® – where map data are now captured and stored in a 'true' vector format.

Even though some advances have been made to modernise the map generalisation process, it still relies on traditional approaches. As a result, manual generalisation continues to take up a significant amount of time and resources, and thus maps derived from large scale products are very expensive to produce. Freeman (1991) describes the process as "*a tedious task, requiring skilled cartographers working for long periods of time*". This problem is further compounded by the need to keep maps up to date. Another considerable disadvantage of manual generalisation is that maps cannot be tailored easily to meet real time, situation dependent needs, such as those required by location based services. Automated map generalisation solutions aim to overcome these limitations.

Traditionally, automated map generalisation research has been carried out within the context of paper-based map production. However, there is a growing demand for maps to be utilised upon a range of digital devices that includes computer monitors, laptops, personal digital assistants (PDA) and the next generation of mobile phones. These devices vary with regards to screen size and display output, ranging from large high

definition TFT screens to lower quality displays such as those found on mobile phones and PDA's. The development of automated generalisation procedures will be of obvious benefit when producing map data that conforms to the limitations of specific screen devices.

1.5 Automated Map Generalisation

The task of map generalisation, traditionally the domain of cartographers, is one of selecting and adjusting the symbols on a map to suit the purpose of the map and the scale of the required output (Robinson *et al.* 1995). The now widespread use of GIS, with their built-in capacity for producing maps, has introduced a requirement to include a facility for map generalisation within these systems. Map production systems use data that are partially pre-generalised, having, in many cases, been derived from cartographic products that are based on a particular scale of representation. However, if a map is displayed at a significantly reduced scale, data objects that were clearly visible as separate entities can become too small to be seen or too close to each other to be distinguishable. In particular, these graphic conflicts arise when the map symbols are no longer a true scale representation of the feature they represent. For example, a road symbol may be much wider, when map scale is taken into account, than the width of the road on the ground (illustrated in Figure 1.2).

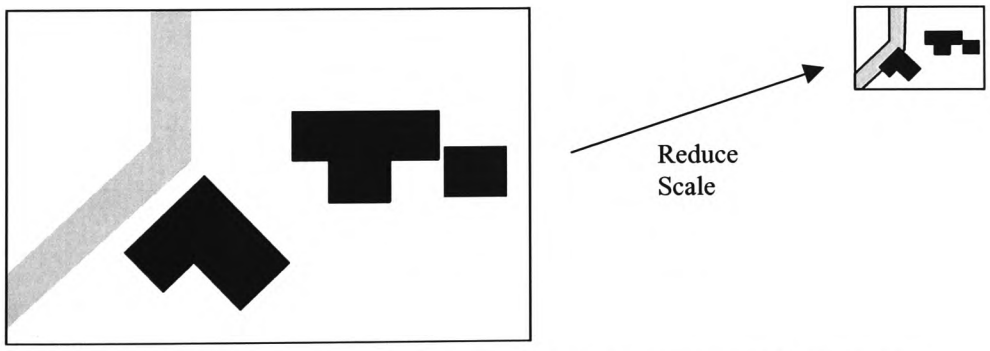


Figure 1.2: Example of conflict between objects subsequent to scale reduction.

A reduction in map scale can lead to many types of deformations, typically in the form of graphic conflict between various map objects. These graphic conflicts arise when the amount of free space in the original map is reduced to a point where objects compete for space. The problem is further compounded when significant map objects (e.g. important

buildings) are enlarged, thus eliminating any remaining free space. The end result is a map that is cluttered, inaccurate and without meaning.

Interest in automated map generalisation has considerably increased over the past decade or so (Buttenfield & McMaster 1991, Muller *et al.* 1995, Weibel 1993, Weibel & Jones 1998). Two such examples include the Automated Generalisation New Technology (AGENT) project (Bader *et al.* 1999; Lamy *et al.* 2000; Haire & Hardy, 2001) and the work on simulated annealing (SA) (Ware and Jones 1998). Details of this and other work are covered in the literature review in Chapter 2.

1.5.1 Advantages and Benefits of Automated Generalisation

Automated map generalisation aims to overcome the limitations associated with that of manual generalisation, and to also enhance and advance upon existing techniques.
A summary of some of the features and benefits that automated map generalisation has to offer are listed as follows (Bundy, 1996):

Cost savings

The generalisation process cannot be considered an easy and simple task. It is in fact quite the opposite, a difficult and rigorous practice. Generalisation is carried out by skilled cartographers requiring many hours labour, most of which can be classed as repetitive, e.g. simplifying and displacing map objects; the cartographer spends a great deal of time performing relatively tedious tasks. Automating the process will drastically reduce the amount of time a cartographer needs to perform generalisation.

Consistency and accuracy

Humans, regardless of intelligence, skill and patience are still vulnerable to tiredness, stress and boredom –resulting in mistakes being made. Humans also lack consistency – making it is possible, given a finite set of rules, for two cartographers to produce similar but in some ways different maps. This is to a large extent due to the different ways in which certain rules can be interpreted.

Increased number of revisions

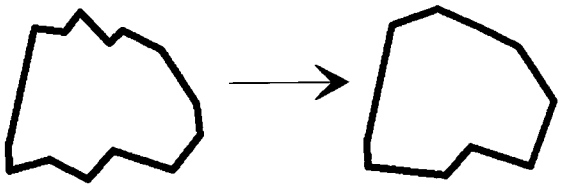
The production of maps is a slow and costly process and this has a considerable effect on the frequency at which maps are revised. Automating the generalisation process will allow changes to a map to be made more quickly and thus the turn-around on revisions can be reduced dramatically.

1.6 Generalisation Operators

Automation of the individual operators required to perform generalisation has been the focus of much work to date. An operator allows various types of transformations to be made upon a single map object at a time or a group of objects simultaneously, assuming the same class of operator is used. Generalisation operators are implemented by means of an algorithm; a classic example is the line simplification algorithm (see Figure 2.1) presented by Douglas and Peucker (1973). This algorithm has attracted much interest over the years and has resulted in similar and advanced operators being developed (e.g. Li and Openshaw 1992, and de Berg *et al.* 1995, 1998). Other generalisation operators, such as displacement, have also received much attention (e.g. Ware and Jones 1998, Nickerson 1988, Ruas 1998 to name just a few).

An outline of the most popular generalisation operators are summarised below (based on a classification given by Shea & McMaster, 1989).

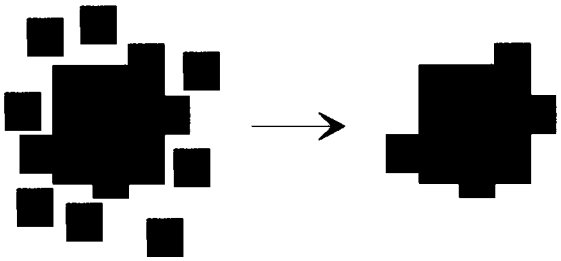
- Simplification of linear features and boundaries (e.g. to reduce the impression of noise in the data). This process involves reducing the number of points in an object, thus making the feature less complex while retaining most of the original shape (see Figure 1.3).



a) Original object. b) Simplified object.

Figure 1.3: Example of simplification.

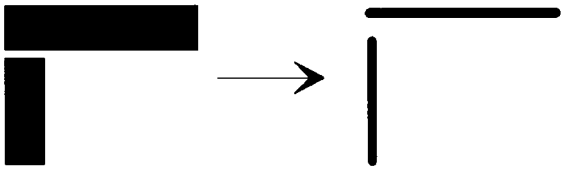
- Elimination of features (e.g. to free-up map space for more important features). This operator removes unimportant features based on a given criteria (such as shape, size, importance etc) but retains more important features (see Figure 1.4).



a) Original object. b) Elimination of features.

Figure 1.4: Example of elimination.

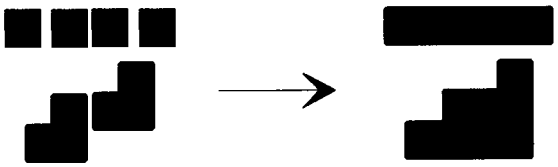
- Collapse in the dimensionality of areal features to lines or points. Features are reduced to a simplified representation (see Figure 1.5).



a) Original object. b) Collapse of features.

Figure 1.5: Example of the collapse operator.

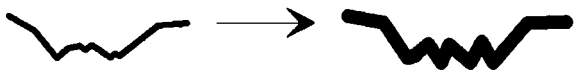
- Amalgamation of adjacent features of the same or similar category to form a new feature (Figure 1.6).



(a) Original object. b) Aggregation of features.

Figure 1.6: Example of the amalgamation operator.

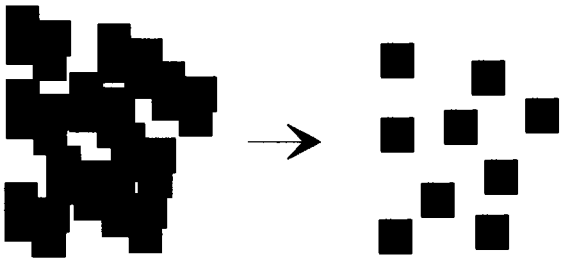
- Exaggeration of important features otherwise too small to represent. In this example the small peaks and troughs in the centre of the line are exaggerated to emphasise their importance (see Figure 1.7)



(a) Original object. (b) After exaggeration.

Figure 1.7: An example of exaggeration.

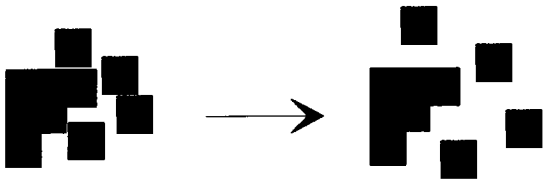
- Typification (or caricature) of the form of features as part of the process of detail reduction. In this example the number of objects is reduced but the overall outline of the original shape exists (see Figure 1.8).



(a) Original feature. (b) Typified feature.

Figure 1.8: Example of typification.

- Displacement of adjacent features that are in graphic conflict with each other. In this example smaller objects are displaced outwardly from the more important feature (see Figure 1.9).



(a) Original. (b) Result of displacement.

Figure 1.9: Example of displacement.

Automation of some of the individual generalisation operators can be found in commercial GIS systems such as ESRI's™ ArcGIS™, Intergraph's Map Generaliser™ and Laser-Scan's™ Lamps2™. Use of these operators however still requires a manual process control. This involves the user in deciding which operators to apply, in which order, and how they should be applied in terms of relevant distance tolerances and other control parameters. Automating the process control is essential if map generalisation is to

graduate from an interactive user-controlled procedure to one that is fully automated (Brassel and Weibel 1998).

1.7 Process Control

Assuming that some, or all, of the basic map generalisation functions are available, there remains the difficult problem of deciding how best to apply them to a specific generalisation request.

The problem is difficult because an effective map is one in which care has been taken to address the interactions between all map symbols, rather than treating the map symbols in isolation. These interactions may give rise to obvious graphic conflicts of proximity and overlap. They may also determine whether important messages, regarding the structure and form of the mapped features, are effectively communicated. The latter interaction could be expressed for example in the alignment of buildings, clustering of woods and lakes, and parallelism between neighbouring rivers and roads. It may be that the problems of graphic conflict can be addressed by a combination of possible actions such as deletion, displacement, aggregation and simplification, combined with appropriate techniques for evaluating the quality of the result. However, a problem with application of individual operators is that each time one of them is applied it may have an effect on a map symbol that was not previously in conflict, resulting in propagation of conflict within the map space. Thus an important aspect of process control is the need for effective strategies for conflict resolution in combination with appropriate quality evaluation.

With this in mind, reference is made to the work of Ruas and Plazanet (1996), who present the idea of a Global Master Plan (GMP) as a means of guiding the map generalisation process. Bundy (1996) suggests something similar to the GMP, referring to what he calls an 'internal agenda'. The AGENT project (e.g. Lamy *et al.* 2000; Haire & Hardy, 2001) makes use of intelligent software agents as a means of controlling the generalisation process. A further examination on previous work related to process control is presented in Chapter 2.

1.8 Summary

Map generalisation is a necessary and critical process in the realm of cartography in order to solve a range of specific problems. This is highlighted in situations when there is a need for scale-reduced maps and when a particular symbology and accuracy of output is required. A majority of the existing generalisation work is still carried out manually by skilled 'human' cartographers. Manual generalisation is a tedious and costly process and is slowly being replaced by newer, automated procedures.

The key to automated generalisation lies within the development and application of effective generalisation operators which perform the underlying geometric transformations of map features. In addition to this we need to rely upon effective strategies to be able to orchestrate them. This is handled in the form of a process control mechanism.

1.9 Aims and Objectives of Research

The main aim of this research is to design, implement and evaluate automated procedures to assist in the process of cartographic map generalisation.

The main research question being addressed is: to what extent can an optimisation technique, such as SA, be used to help automate the map generalisation process?

In order to answer this question, the work contained within this thesis builds upon the work of Ware and Jones described in their paper, "Conflict reduction in map generalisation using iterative improvement" (Ware *et al.* 1988). For the benefit of the reader, a summary of this work is presented in Chapter 3.

The aims of the research can be summarised as follows:

1. To carry out a thorough evaluation of the existing SA technique.
2. To assess the suitability of SA as a means of a process control to assist in cartographic generalisation.
3. Design, implement and evaluate new techniques to overcome the limitations of SA and integrate these into a new generalisation framework.

The research makes use of Ordnance Survey® (OS™) large-scale topographic data. OS™ MasterMap® provides the necessary areal features such as buildings, while OS™ OSCAR® provides road centreline data. The research carried out in this project conforms to generalisation specifications (i.e. minimum separating distances between objects) as suggested by the OS™.

The valuable outcome of this research will result in software that will allow certain generalisation tasks to be carried out automatically.

Initial research uncovered a series of limitations and problems with the original SA process. These are:

1. **Slow execution times.**

   Initial results showed that the execution times of the original SA proved too slow for it to be used in various applications (e.g. large data sets and location based services, where map data needs to be served to the user in a short amount of time).

2. **Limited support of generalisation operators.**

   The current implementation only makes use of a single operator, i.e. displacement. This does not guarantee the removal of all graphic conflict.

3. **Problems with maintaining high order feature alignment.**

   The original SA system was effective in reducing a large proportion of graphic conflict. However little regard was given to spatial relationships. As a result the algorithm tended to produce maps in which important spatial relationship between objects, which existed in most, if not all maps, were broken (e.g. misalignment of buildings). The resolution of this issue was a significant requirement of this project.

4. **Lack of supported feature classes.**

   The original algorithm only dealt with two feature classes, one line theme and a polygon theme (usually a road network theme and a building polygon theme). The requirements set out by the OS™ required a system that would handle additional feature classes such as ponds, forests, railway lines and rivers, each having their own independent generalisation constraints (i.e. minimum separating distances between each feature class).

The research objectives are to provide solutions to the above problems, particularly in the direction of tackling the problems associated with graphic conflict reduction.

1.10 Outline to the Thesis

Chapter 1 provides background and rationale for the research including a brief overview of digital cartographic generalisation, which introduces important concepts and principles that underlie the contents of this thesis.

Chapter 2 provides a literature review of automated generalisation research, emphasising the particular importance of displacement and optimisation techniques.

Chapter 3 describes previous work on SA carried out at the University of Glamorgan.

Chapters 4, 5, 6, detail the additions and advancements that were made to extend the original SA technique.

Chapters 7&8 concludes the thesis with an overview of the results and an examination of the contribution of research. A discussion on potential future work is also included.

Chapter 2

Literature Review

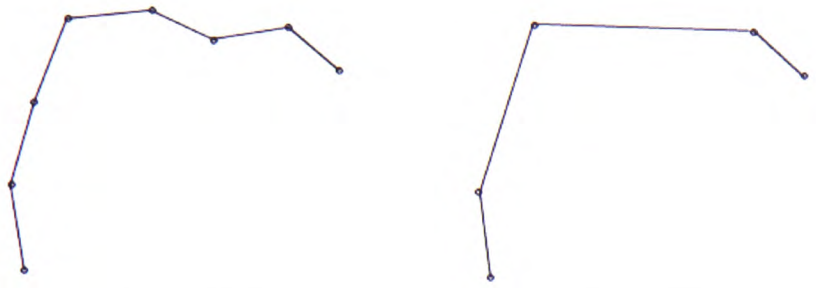
This chapter contains a review of relevant literature related to automated map generalisation, concentrating in particular on displacement, process control and optimisation techniques. This review is not intended to give an exhaustive account, but instead provide the reader with a flavour of the research that has been achieved in the field, highlighting particular strengths and weaknesses where considered appropriate.

2.1 Introduction

The chapter starts with a concise overview of the pioneering work on generalisation associated with linear features (e.g. roads, coastlines etc) including the well-known Douglas-Peucker algorithm for line simplification. Following on from this is an outline of achievements in the area of areal (polygonal) generalisation (such as building typification). The chapter continues by outlining previous research on displacement procedures and provides work related to combining multiple generalisation operators. The chapter concludes with an overview on the past efforts related to optimisation techniques.

2.2 Line Generalisation

Early attempts during the 1970s at automating the map generalisation process were primarily concerned with the simplification of linear features. Line simplification can be defined as reducing the number of points used to represent a line whilst retaining its salient features of character. The best known and most widely used algorithm is Douglas-Peucker (Douglas & Peucker 1973). Intended originally to be used for point filtering, it works on the principle of selecting important points only (see Figure 2.1). This algorithm being best suited for minimal line simplification and fails to provide a solution for complex line generalisation; as a result several subsequent algorithms have been developed (e.g. Chrobak 2000, Saalfeld 1999).



(a) Original line. (b) Simplified line.

Figure 2.1: Douglas-Peucker algorithm (1973).

Irregular linear features, such as non-urban roads, rivers and coastline, have also been dealt with extensively (e.g. Visvalingam and Whyatt 1993, Christensen 1999). Visvalingam and Whyatt (1993) overcome the limitations of the Douglas-Peucker algorithm (Douglas & Peucker 1973) by achieving minimal simplification and caricatural generalisation. Some algorithms have taken into account the more advanced issues of symbolisation and context. For example, Lecordix *et al.* (1997) provide a number of promising solutions (to be used interactively with the classic line simplification and smoothing algorithms) that deal with the problem of line coalescence caused by symbol widening. For coastlines, the work of Wang and Muller (1993) makes use of both procedural and logic programming. Initially insignificant features are identified and removed, then features that remain are generalised; rivers are widened using a buffering technique and coastlines are simplified using the Douglas-Peucker algorithm.

Van der Poorten *et al.* (2002) explores an alternative approach to the problem of performing line and polygon generalisation by elimination of discernible features according to shape criteria; this technique makes use of constrained Delaunay triangulation (De Floriani and Puppo 1998, Chew 1989) and offers some potential advantages over existing methods (e.g. where the topological relations between linear objects are preserved). Wang and Muller (1998) describe a technique based on shape analysis for generalising a single line using exaggeration, combination and elimination of bends (this algorithm forms the basis of ESRI's™ BendSimplify operator). The experimental system makes successful use of this shape bend analysis technique, although several limitations are known to exist, especially when dealing with more complex linear features. Mackaness (1995) has made good progress by use of alpha analysis for classifying urban road networks hierarchically, providing a means for removing roads at a

smaller scale while still conveying essential characteristics of the network. The algorithm can be applied to any street network without requiring additional attribute information. However a drawback to his approach is when this attribute information is made available, the algorithm fails to make sufficient use of it.

Several algorithms address specifically the issue of topological correctness. For example, Lichtner (1979) describes a process that takes account of the dimensions of protruding parts of features, while de Berg *et al.* (1998) makes use of a quadratic time algorithm that maintains topological relations and tackles the problem of reducing the storage space required by complex linear features. His approach demonstrates how the Douglas-Peucker algorithm (1973) can be extended and offers several options for speeding up the original algorithm.

A common failing present in many of the linear generalisation methods summarised in this review is that no single line simplification approach offers a comprehensive solution.

2.3 Areal Generalisation

The development of generalisation algorithms to act upon areal features such as buildings, ponds, lakes etc, has also received much attention. Automation in generalising areal features may include amalgamation (where small objects are merged to form a larger representation) and elimination (the removal of smaller, non-important features). More significant and extensive, has been the development and application of displacement procedures to solve various forms of graphic conflict. A detailed account of this work is provided in section 2.3.1.

There has been some exploratory work carried out in the area of building typification. Sester (2003) and Moulin (2003) both present methods that make use of Kohonen Self Organizing Maps (Kohonen 1982). Regnauld (1996, 2001) presents ideas relating to building typification that creates a result with fewer objects, and preserves the initial pattern of distribution. The process of carefully selecting which objects should be retained in the final display is achieved using a proximity graph, which is analysed and segmented to various criteria based on the Gestalt theory. Buildings are grouped together prior to typification using Minimum Spanning Trees (MST) (see Figure 2.2) which appears to work well (Zahn 1971).

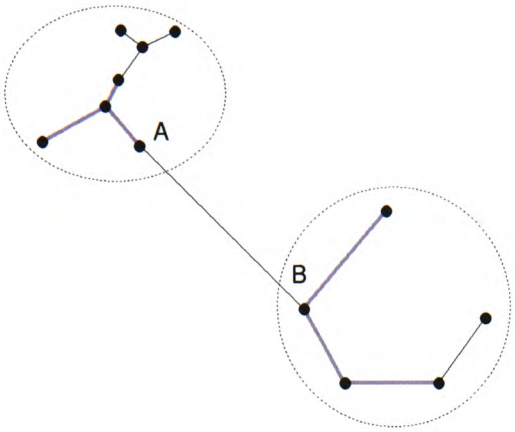


Figure 2.2: Segmentation of a MST resulting in two distinct regions, A and B (Regnauld 1996).

Ai and van Oosterom (2002) introduced a field based method inspired by Ruas (1998) and Højolt (2000) that make use of a triangulated data structure to assist in calculating displacement vectors, which are applied to certain features prior to amalgamation. The research addresses the practical problem caused by street widening and offers a solution that makes use of a displacement field model improving on early attempts by Højolt (2000). The results and effectiveness of this research appear inconclusive due to the limited experiments and evaluation presented.

DeLucia and Black (1987) propose a number of triangulation-based area amalgamation procedures for both man-made and naturally occurring features. These ideas are taken up and advanced in Jones *et al.* (1995). An alternative aggregation method using a series of morphological operators has been presented by Su *et al.* (1997) (this, or something similar, forms the basis of ESRI's™ AreaAggregate function). Results published by the author suggest these methods appear to work well, even though the research only deals with algebraic models that operate at a very basic level. It would be interesting to see the application of these morphological operators to larger, more detailed raster-based data sets.

Map features can be enlarged by applying a simple scaling to each of its vertices. The technique of Jones *et al.* (1995) advances this by removing the need for a fixed anchor point, and by providing an automatic conflict check. Creating a caricature of a map feature is more difficult. Sester (2003) claims that her technique can achieve something

similar, but provides very little detail. The main problem is locating the specific parts of a feature deemed important enough to warrant exaggeration.

Pattern and context analysis techniques have been developed for use as part of other algorithms. Mackaness presents many useful ideas on how to cluster features for the purpose of map generalisation (Mackaness 1994, Ormsby and Mackaness 1999, Mackaness and Mackechnie 1999). For polygonal objects, and buildings in particular, techniques based on the use of MST's appear to work well (e.g. Regnauld 1996, Bader and Weibel 2003). Triangulating between features has been used as a means of digitally representing the white space on a map, and these triangles prove useful in calculating the space in between map features (e.g. Jones *et al.* 1995, Ai and van Oosterom 2002).

2.3.1 Displacement Techniques

The displacement of map features (especially in urban areas) to resolve spatial conflict has been the subject of much research (Lichtner 1979, Nickerson 1988, and Monmonier, 1986, 1987, 1991). Recent work has seen the development of supplementary algorithms and procedures (Mackaness 1994, Ruas 1998, Bader and Weibel 2003).

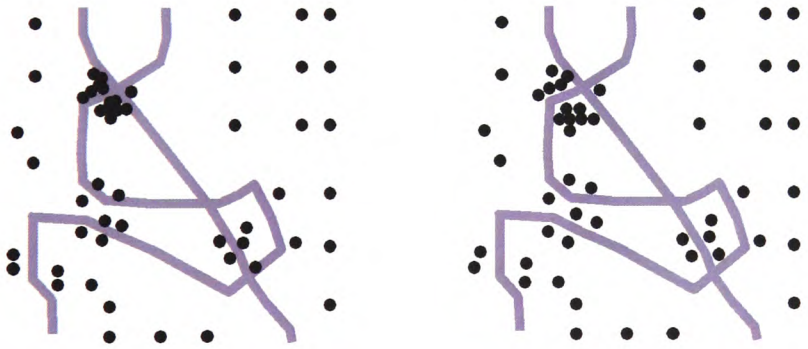
Lichtner (1979) addressed problems associated with the symbolisation of linear features. His solution was achieved by displacing a map feature in and around regions of conflict at right angles to the axis of the symbolised line, the distance of displacement being decreased linearly from a maximum distance (for features nearest to the conflict) to zero (for features furthest away). Although Lichtner's approach reduced the amount of conflict, map features became geometrically distorted (as vertices were displaced that constituted the make-up of the polygon). This however was intentional as the spatial accuracy of map features were given a higher-level of precedence over that of distortion.

Nickerson (1988) developed a complicated constraint-based system as a means of displacing map features. His Mapex system consisted of a three-stage process that guided displacement using a series of constraints. These constraints were to: i) maintain topology and connectivity; ii) maintain the shape of the feature as much as possible; iii) displace features as little as possible. His system also consisted of English-type rules which were used to specify what map features were to be eliminated at the reduced scale. Linear

features were also subjected to various forms of polyline simplification, while conflict resolution was achieved through interference detection and by displacement propagation.

Monmonier (1986, 1989 and 1991) attempted to utilise pre-generalised maps as a guide for determining distances and directions for the displacement of features that appeared on the source and guide map. This approach initially appeared promising, but later proved ineffective. One problem was caused by trying to match features between the source and guide map, as features had become subjected to various forms of areal transformations.

Mackaness (1994) used a modified radial displacement function (MPRD) (see Figure 2.3) to resolve proximity conflicts between points, and subsequently on other features such as buildings. Algorithms are used to first detect the source of the conflict and then cluster analysis ascertains all remaining objects that also contribute to the conflict. Map features (vertices) are then displaced in a radial manner away from the source of the conflict. This approach has the advantage that all objects in conflict, share the effects of displacement thus limiting the overall amount of spatial disturbance. However the use of a radial displacement function is not entirely suited for displacement techniques as shape distortion can occur.



(a) Before. (b) After.

Figure 2.3: Application of MPRD (Mackaness 1994).

Li and Su (1996) developed a mathematical model for solving problems associated with translation and modification. This mathematical model consists of morphological operators that physically change the structure of map features through the use of dilation and erosion. This work however is limited for use on raster-based data sets and offers little practical solutions for vector-based data.

Ruas' (1998) approach to displacing map features makes use of a reactive displacement method to analyse spatial relationships between various objects to determine if displacement is permissible in the first instance, and whether the outcome of the final solution is an acceptable one. The key to this process relies upon the declaration of a set of constraints and a mechanism of evaluation to control the displacement process. This mechanism consists of twelve-stages that are executed in a sequential manner therefore allowing for the initial removal of buildings and roads, and building aggregation based on density analysis.

Harrie (2000) attempts to automate the displacement of vector data so that features remain distinguishable after scale reduction has occurred. Harrie adopts a similar strategy to that of Ruas (1998) in which constraints are used to guide the generalisation process. Harrie's approach makes use of the Least Squares Method (LSM). The main function of the LSM is concerned with deriving a compromise between conflicting constraints (e.g. objects should move as little as possible, no spatial conflicts are allowed). The experiments make use of 1:10,000 scale data sets comprised of polygons and lines (roads and buildings) and results achieved showed the conflict was resolved. However, Harrie does admit that his approach is slow and also his method does not deal with self-coalescence (see Figure 2.4).

Figure 2.4: Application of LSM (Harrie 2000) in which buildings are displaced away from their original location (shown in grey) to resolve graphic conflict between a road feature.

Lonergan and Jones (2001) make use of a quality measure in terms of minimum distance violations. This measure is combined with an iterative improvement technique based on maximising nearest neighbour distances. Their experimental evaluation compares the effectiveness of this approach against one which uses simulated annealing. Results overall remain inconclusive, although this approach has reduced the amount of graphic conflict.

Recent work by Bader and Weibel (2001) has addressed the problems associated with the displacement and deformation of linear features such as roads (see Figure 2.5). Their approach makes use of an energy-minimising spline called a 'snake'. These snakes are guided by internal constraint forces and influenced by external enforcement. The use of these energy minimising splines originate from the field of pattern recognition and computer vision (Kass *et al.* 1987) and was adapted for use in displacement by Burghardt and Meier (1997). The process of displacement using this 'snakes' technique relies upon the structure of the spline itself, which is composed of internal and external energy. The forces acting upon this energy allows the natural differentiation of propagation and displacement to take place. Results published on the use of 'snakes' to resolve conflict between linear features look extremely promising. Bader concludes in his research that his approach exceeds the cartographic quality achieved by previous known algorithms. However he does mention that his technique also has a few limitations such as a lack of accuracy when separating objects to a minimal distance.

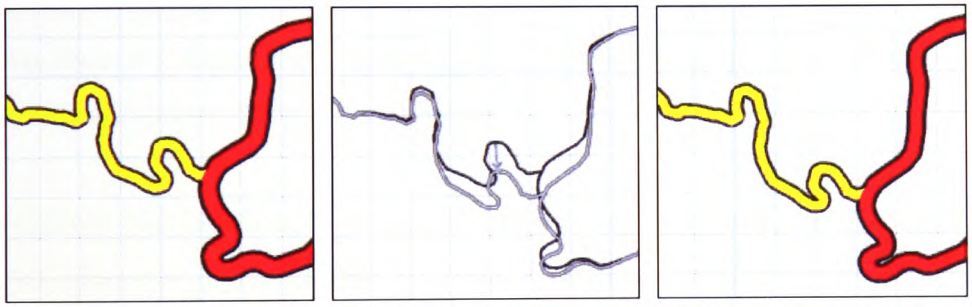


Figure 2.5: Snakes model (Bader *et al.* 2001). Original map (left); application of 'snakes' (middle) in which part of a linear feature is distorted; output map (right).

Galanda (2001) continues the work on 'snakes' (Burghardt and Meier 1997, Bader 2001) and establishes a holistic framework in order to perform polygon generalisation. The original implementation on the use of 'snakes' to perform linear generalisation is adapted to allow for displacement, enlargement and exaggeration of polygonal features such as buildings (or, to be precise, the lines that make up a polygon) to maintain legibility. The algorithm allows for the simultaneous resolution of size and proximity conflicts while avoiding new conflicts being introduced during the generalisation process. Results published demonstrates the overwhelming potential of applying energy minimising techniques for polygon generalisation

The work of Li *et al.* (2002) present a web-based map information retrieval system called MAPBOT. Polygons are displaced using a two-level Agent-based architecture. At the highest level, a so-called house keeping agent takes overall control and monitors the overall condition of the map. At a lower level, each map feature is assigned its own map agent that performs a check on the environment (such as its neighbours and in terms of conflict). Next a map agent sends requests to other objects (e.g. move) and finds appropriate solutions by deciding on which request to send and whether or not to move itself. The results of this research show potential and are more promising than the work of Ware and Jones (1998) and Lonergan and Jones (2001), both in terms of conflict reduction and execution times.

2.4 Orchestration of Multiple Operators

It has long been recognised in the field of automated cartography that the use of a single operator such as displacement or simplification in isolation is not sufficient for the removal of all graphic conflict. Therefore in addition to the development of various operators there is a requirement to develop some means of process control to be able to productively administer them in combination.

To continue in this direction, some authors suggest controlling the generalisation process by use of rule-bases (e.g. Shea 1991). A problem with adopting a rule-based approach is that some situations will require strategies that are very difficult to represent as an if-then-else rule since some courses of action cannot be entirely predetermined because they depend upon the interaction between multiple map features.

Ruas and Plazanet (1996), present the idea of a GMP as a means of guiding the map generalisation process. The GMP is an ordered list of the actions required for generalisation. The plan is defined according to: i) database specifications, ii) knowledge of the logical ordering of actions and iii) an initial understanding of the data being generalised. Bundy (1996) suggests something similar to the GMP, referring to what he calls an 'internal agenda'. This agenda is a broad description of the processes of generalisation, defining the sequence in which generalisation operators are to be applied.

Lichtner (1979) was one of the first researchers to combine multiple operators and an early form of process control, in which he made use of 'batch-mode' generalisation that involved the use of a selection of contours and other linear features. The selection and enlargement of buildings and the aggregation of buildings features avoided the need for processes to be repeated and the need for frequent correction by interactive means. The success of this early work led to the generalisation of a small data set from 1:5,000 scale to 1:25,000.

Ai and van Oosterom (2002) combines displacement and aggregation in an iterative sequential manner. Their simple progressive approach attempts to solve spatial conflict by firstly displacing map features, and in the event that displacement is unsuccessful, they subsequently make use of the aggregate operator to merge buildings together that lie-within a specified distance of each other. The process is then repeated making use of displacement operator once again.

Harrie *et al.* (2002) resolves graphic conflict that arises as a consequence of scale reduction by combining various operators. Smoothing, simplification, exaggeration and displacement algorithms are used on linear and areal features. The approach makes use of a series of analytical constraints along with finite rules to dictate when these constraints should be best applied. Use is made of the Least Squares Adjustment Method (LSM) to find the optimal solution to these constraints. A system has been developed in C++ that successfully demonstrates the usefulness of the approach in three experiments. The first experiment makes use of line simplification, polygon exaggeration and line displacement. The second test makes use of a small 1:10,000 vector data set that is successfully generalised to a reduced scale of 1:50,000. The final experiment demonstrates the performance and effectiveness of his algorithm.

Other attempts have been made to combine the use of multiple operators to perform generalisation. The most ambitious effort was made by the AGENT project (see Figure 2.6). The AGENT project utilised agent-based methodologies as a means of providing support for the derivation of multi-scaled products from a single detailed database with the least amount of human intervention.

Figure 2.6: Example of agent-based methodologies (AGENT Project). A squaring transformation is applied to an object (left); the smallest parts of the object are widened (middle); final result (right).

2.5 Optimisation Approaches

Various researchers have turned towards physics and mathematics for modelling and developing optimisation solutions to solve a range of generalisation problems including displacement (Bobrich 1996, Bader and Weibel 2001, Ware and Jones 1998).

Bobrich (1996) was the first to make use of an optimisation algorithm that incorporated displacement. His work was based on the concept of 'springs' which constituted various degrees of stiffness (this modelled different constraints) for displacing road features (see Figure 2.7). Displacement was performed using a raster-based displacement potential, while optimisation was performed by a Downhill-Simplex algorithm (Nelder and Mead, 1965) which negotiated a balance in terms of forces between the springs and the displacement potential.

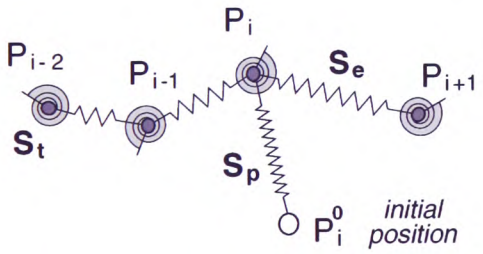


Figure 2.7: Spring model. Each line represents various degrees of stiffness (Bobrich 1996).

Ware and Jones (1998) addressed the problems of graphic conflict between buildings and between buildings and roads using fixed trial positions (as possible displaced-locations) in combination with iterative improvement algorithms such as gradient descent and simulated annealing. The use of these algorithms in conjunction with an effective cost function significantly reduces the number of map realisations that are generated and evaluated to produce a near-optimum solution. A more thorough description of this approach is given in Chapter 3.

Højolt (2000) proposes a finite element method and treats the map space as an elastic body. In his approach, objects that are subjected to change are allocated a force which expands and repulses surrounding objects outwardly. To prevent this type of deformation affecting more important objects, each one is allocated a predefined stiffness. A balance is then calculated (making use of the finite element method) between the forces.

Bader and Weibel (2003) present an optimisation approach in which a truss structure is used to encapsulate important spatial relations between the polygons. This truss can be thought of as being made up of elastic beams, modelled as finite elements, connecting polygon centroids. The truss is constructed using a MST. Minimum clearance violations give rise to forces being exerted at truss nodes; the stiffness of a particular truss rod is related to the strength of relationship between the two buildings it connects. The algorithm works so as to deform the truss iteratively by applying the principle of energy minimisation, hence resolving conflict while striving to maintain important relationships.

Sester (2003) proposes an optimisation solution by combining LSA and neural networks. The LSA approach makes use of a functional model together with the use of a stochastic model to generalise by means of simplifying building ground plans and displacing various cartographic objects.

2.6 Summary & Conclusions

This chapter charts significant progress in the development of automated map generalisation. The work on linear features and, most notably, the use of line simplification formed the early research into automating the cartographic generalisation process. Several algorithms challenge specifically the issue of topological correctness while pattern and context analysis techniques have received little attention.

A significant proportion of the research on generalisation to-date has focused on the development of various feature displacement algorithms and optimisation techniques for linear and areal features. Displacement of areal features such as buildings has received the most considerable effort. Those investigated in this review all offer promising results (e.g. Mackaness 1994, Ruas 1999, Harrie 2000); however, in each case, the evidence is inconclusive due to the limited nature of the data sets used.

Alternative methods to solve automated generalisation have also relied on the use of multiple operators and the use of process control to direct them. The most ambitious effort to-date to combine various operators was undertaken by the AGENT Project which utilised agent-based methodologies.

Other types of research efforts have led to the development of various optimisation techniques, for example the work on SA by Ware and Jones (1998) Although these optimisation techniques succeed in resolving graphic conflict, they do suffer from a number of limitations. One example is their inability to preserve spatial relationships (as buildings become misaligned during the displacement process). It is evident that further work is needed in this field.

Chapter 3

Graphic Conflict Reduction Using Simulated Annealing

This chapter outlines previous work on SA carried out at the University of Glamorgan.

3.1 Introduction

The previous chapter introduced various techniques that have been developed to tackle the problems associated with automated cartographic generalisation. The aim of this research is to extend and advance the SA work of Ware and Jones (1998) and to determine whether or not it can be used as a generalisation tool for use in a broader automated generalisation framework. By way of an introduction this chapter presents the reader with background information on iterative improvement techniques (e.g. gradient descent and SA) and an account of the existing SA work.

3.2 Map Generalisation Using Iterative Improvement

3.2.1 Trial Positions

One approach to resolving graphic conflict caused by scale reduction is to displace map features. In the work of Ware and Jones (1998), pre-computed trial positions are used to resolve graphic conflict. This approach is adopted from Christensen *et al.* (1995) where trial positions are used to solve map labelling problems, and is probably best introduced in this context.

Consider a simple example point data set, where each record consists of an x, y co-ordinate pair and a single text item; each record represents the location and name of a town. If the data set were to be displayed graphically, then some decision has to be made as to where to position each text item in relation to its associated point. A simple solution is to position text in some default position. For example each label could be positioned above and to the right of its associated point (see Figure 3.1).

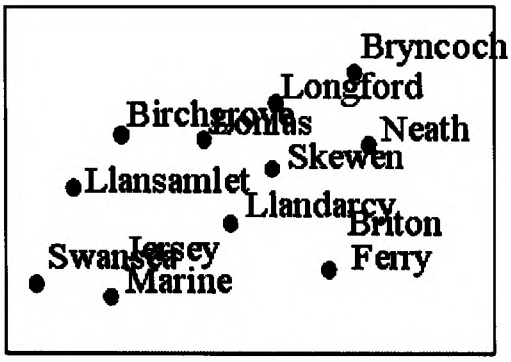


Figure 3.1: Sample label data set.

The obvious problem with this approach is that it is likely to produce a very poor map in which graphic conflict occurs (i.e. labels that overlap other labels and labels that overlap points); a further problem is that it is sometimes difficult to establish the correct associations between points and labels.

A tried and tested approach to solving the point feature map labelling problem is that of trial positions (e.g. Christensen *et al.* 1995). The idea is to assign each point feature a number of pre-defined trial positions, each of which represents a valid label location (Figure 3.2).

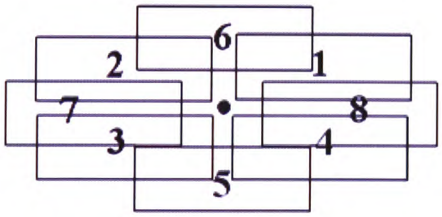


Figure 3.2: Trial position strategy.

If there are *'n'* point features, and each is assigned *'k'* label trial positions, then there are a total of k^n alternative map labellings available; the assumption is that some of the labellings provide better solutions (i.e. less graphic conflict) than others (e.g. see Figure 3.3).

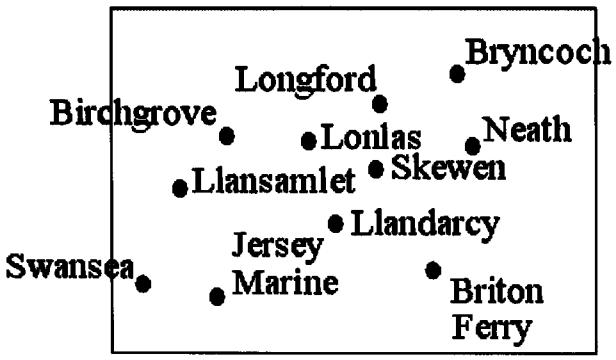


Figure 3.3: Point feature label placement using trial positions.

The label placement problem now becomes an optimisation problem (i.e. finding a labelling with least conflict). Generating and evaluating all realisations is not practical, even for relatively small values of n and k. Furthermore, Marks and Shieber (1993) have shown that this problem is NP-hard (i.e. it is unlikely that an optimal solution can be found in polynomial time). In response to this a number of approximation algorithms (i.e. algorithms that attempt to find near optimal solutions) have been proposed in the literature; examples include discrete gradient descent (Christensen *et al.* 1995), simulated annealing (Christensen *et al.* 1995, Zoraster 1997), tabu search (Yamamoto *et al.* 2002) and genetic algorithms (van Dijk *et al.* 1998).

3.3 Graphic Feature Displacement Using SA

Ware and Jones (1998) tackle the problem of graphic conflict reduction by displacing polygon map features (e.g. buildings). They consider the specific problem of polygons that interfere either with each other or with fixed linear map features (e.g. roads). The method they present is an adaptation of the trial position combined with the SA approach given by Christensen *et al.* (1995); the major difference is that it is polygonal map features, not labels, that are assigned trial positions (and subsequently displaced).

The previous work by Ware and Jones (1998) is now summarised.

3.3.1 Object States

To continue the discussion on trial positions, consider a map display D, which is made up of fixed linear objects F, and n modifiable detached polygonal objects, M. Each

modifiable object $m_i \in M$, has k possible states, providing a total of k^n possible configurations of D.

At any given time a particular object, m_i, exists in one of its k states (or trial positions). An object's initial map position is designated as being trial position 1. It is assumed that during the course of generalisation, each object m_i can be displaced up to some maximum distance d, from its original position (i.e. there is a continuous space that extends from m_i by a distance d into which it is permissible for m_i to move). The displacement trial positions associated with m_i represent a discrete approximation to this continuous space. Each object m_i has k displacement trial positions, which are distributed evenly about the object (see Figure 3.4).

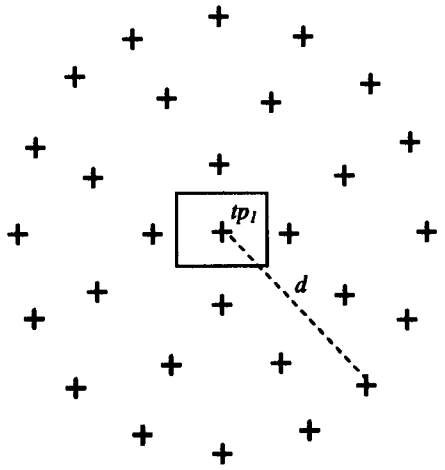


Figure 3.4: Example of a graphical object with 28 displacement trial positions.

3.3.2 Evaluation of Map Display

For a particular configuration D_j, each object m_i has an associated cost. The cost is determined by the extent to which m_i is in conflict. Two categories of spatial conflict are considered:

(i) Type-1 conflict between a pair of polygonal objects (i.e. m_i and m_j interfere with each other). This conflict occurs when the minimum separating distance (in viewing coordinates) between m_i and m_j is less than some predefined threshold d_{min1}. An occurrence of this type of conflict carries a cost c_1.

(ii) Type-2 conflict between a polygonal object and a linear object (i.e. m_i and $f_j \in F$ interfere with each other). This conflict occurs when the minimum separating distance (in viewing coordinates) between m_i and f_j is less than some predefined threshold d_{min2}. An occurrence of this type of conflict carries a cost c_2.

The total cost $\mathbf{C}(D_j)$ associated with a realization D_j is found by summing the costs associated with each object $m_i \in M$. The goal is to find a minimum cost configuration D_{min} such that:

$$\mathbf{C}(D_{min}) = \mathbf{MIN}(\mathbf{C}(D_j), j=1,2\ldots k^n).$$

The set of all configurations is referred to as the search space. If the search space is small enough then D_{min} can be found by generating and evaluating each configuration D_j ($j=1,2\ldots k^n$) in turn. However, this is not practical for realistic values of n and k. For example, a relatively simple display consisting of 10 modifiable objects, each with eight trial positions gives rise to approximately 1,000,000,000 configurations. Results reported later in this chapter suggest it would take approximately 20 hours to process this many configurations (generating and evaluating a single configuration takes approximately 0.00008s).

3.3.3 An Iterative Improvement Solution

A well-established approach to solving large optimization problems of the kind described is to adopt an iterative improvement algorithm (a special class of approximation algorithm). These algorithms belong to a sub-class of heuristic algorithms whose sole purpose is to provide near-optimal solutions to complex and sophisticated problems.

These algorithms work by searching a given space starting from an initial solution and making use of a mechanism to progress from one solution to another. The objective is to spawn multiple neighbours of the current solution in an attempt to find a solution that has a lower cost than the current one. The concept of iterative improvement can be illustrated by considering the search space to be laid out on the surface of a landscape. The elevation at any point on the landscape represents an evaluation cost for the particular solution associated with that point. An iterative improvement algorithm will move around the landscape in an attempt to find the lowest troughs, which correspond to low cost solutions (Russell and Norvig 1995). Two examples of iterative improvement algorithms are discrete gradient descent and SA.

3.3.3.1 Graphic Conflict Reduction Using Gradient Descent

```
function GradientDescent
    input: D_initial
        while D_current←D_initial
            D_new←LowestCostSuccessor(D_current)
            if C(D_new) ≥C(D_current) then Return(D_current)
            D_current←D_new
        end
Return(D_current)
```

Figure 3.5: Gradient descent algorithm.

Figure 3.5 describes a simple gradient descent implementation. The algorithm accepts an initial map configuration $D_{initial}$ (i.e. each object in trial position 1), which is immediately designated as being the current solution $D_{current}$. Next the lowest cost successor D_{new} to $D_{current}$ is found. A particular successor to $D_{current}$ is found by moving a single object m_i to an alternative trial position; the lowest cost successor can be found by generating and evaluating all possible successors (of which there are k^{n-1}). If D_{new} represents an

improvement on $D_{current}$, then D_{new} becomes $D_{current}$ and the next lowest cost successor is generated. This process is repeated until a D_{new} is generated that offers no improvement; at this stage the algorithm terminates, with $D_{current}$ being returned as the solution. The algorithm is quite straightforward, but is not guaranteed to find an optimal solution, since it is possible to arrive at a non-optimal current state from which no better state can be reached. This occurs when the search descends into a local minimum, from which any single displacement generates a worse state. To use the landscape analogy once more, a local minimum can be thought of as a trough in the landscape that happens to be higher than the lowest point on the landscape. Several ways of trying to deal with the problem of local minima are available (e.g. random-restart, backtracking and multiple-moves). However, given the exponential nature (as the number of map objects increases, the number of possible map configurations grows exponentially) of most realistic search spaces makes such remedies impractical.

3.3.3.2 Graphic Conflict Reduction Using SA

The origins of SA can be traced back to the foundations of thermodynamics where liquids solidify by forming crystalline structures during the cooling process. Blacksmiths in the Iron Age discovered that the slower the rate at which a metal is cooled the more perfect the crystals formed. This is due to the randomness of the thermal energy which allows atoms to escape from local minima.

During the 1950s, Metropolis developed an algorithm which simulated this thermodynamic phenomena. His method models the metal as a system of particles, in-which the algorithm simulates the cooling process by gradually lowering the temperature of the system until it reaches a frozen state (Metropolis *et al.* 1953). Later, Kirkpatrick improved upon Metropolis' technique and applied his algorithm to solve a range of optimisation problems (Kirkpatrick *et al.* 1982, 1983). Since then the use of SA as an optimisation solution has drawn the attention of many authors inspired by Kirkpatrick's efforts (Rutenbar 1989, Ingber 1993, Dowsland 1995, Zoraster 1997, Ware and Jones 1998).

SA is similar to gradient descent in that during the course of algorithm execution a new solution is always accepted provided it offers an improved result. The problem of getting

caught in local minima (as found with gradient descent) is overcome by sometimes allowing the current solution to get worse (that is, to allow uphill moves to be made). During the running of the algorithm the probability of accepting a worse state is decreased over time until no further worse states are accepted. SA then behaves in the same manner as that of gradient descent. The difference between these two algorithms is illustrated in Figure 3.6.

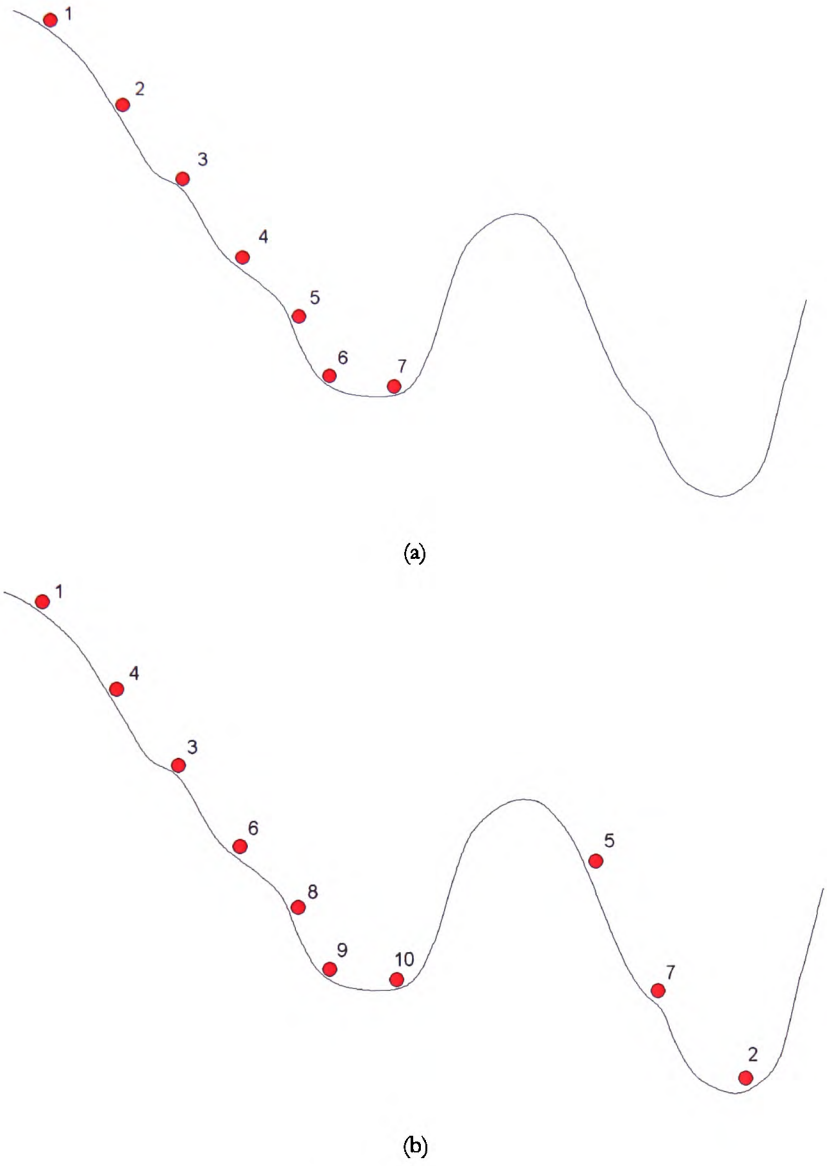


Figure 3.6: Schematic illustration of gradient descent (a) and SA (b).

The above schematic diagram shows a ball travelling along a landscape. With gradient descent the ball traverses the landscape and accepts lower-cost solutions thus moving in a downhill direction. Inevitably the ball will reach a point where it will encounter an

uphill barrier (representing a higher cost solution) and therefore stops (local minima). In SA the ball is capable of traversing a rougher landscape where it encounters several peaks (higher cost solutions) and troughs (lower cost solutions) where occasional moves in the wrong direction are accepted. This prevents the ball from becoming trapped in a local minimum.

A plot of a typical annealing schedule (see Figure 3.7) shows that at high temperatures many higher cost successor solutions are accepted, while when lower temperatures are encountered fewer worse successors are accepted. Furthermore by analysing the graph, the effectiveness of SA can be seen in that most of the work is achieved during the early-stages of the annealing schedule (represented at the higher-temperature range of the graph).

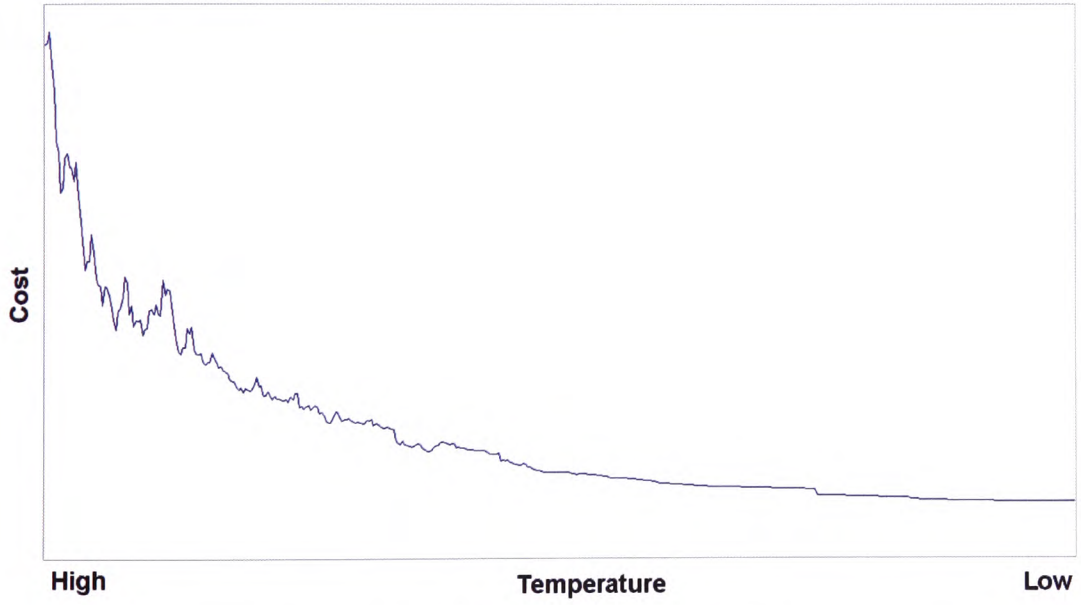


Figure 3.7: Plot of a typical annealing schedule.

```
function SimulatedAnnealing
        input: D_initial, Schedule, StopConditions
        D_current←D_initial
        T←initialT(Schedule)
        while NotMet(StopConditions)
                D_new←RandomSuccessor(D_current)
                ΔE←C(D_current)-C(D_new)
                if ΔE >0 then D_current←D_new
                else
                        P = e^(-ΔE/T)
                        R=Random(0,1)
                        if (R<P) then D_current←D_new
                end
                T←UpdateT(Schedule)
        end
Return(D_current)
```

Figure 3.8: SA algorithm.

Searches based on SA (Kirkpatrick *et al.* 1983) attempt to overcome the problem of getting caught in local minima by allowing some non-improving configurations to be accepted. As with gradient descent, SA (see Figure 3.8) always accepts D_{new} if it offers a better solution than $D_{current}$. However, in cases where D_{new} provides no improvement, SA will accept the new configuration with some probability P (P<1). Like gradient descent, the algorithm begins by accepting an initial map configuration, $D_{initial}$ (i.e. each object in trial position 1); this is immediately designated as being the current solution, $D_{current}$. Next a random successor, D_{new} , is generated by moving a randomly chosen object m_i to a randomly chosen trial positions k_j.

If the displacement results in a display configuration with a lower cost, $\mathbf{C}(D_{new}) < \mathbf{C}(D_{current})$, then the object remains in the chosen trial position ($D_{current} \leftarrow D_{new}$). If, however, the new display has a higher or equal cost, i.e. $\mathbf{C}(D_{new}) \geq \mathbf{C}(D_{current})$, then the object is either returned to its previous position or remains in its new position, depending on probability P. The process of attempting a random object displacement continues until stop conditions are met (e.g. a solution which meets a target cost is found or a pre-defined maximum number of iterations have taken place or a pre-defined maximum amount of time has elapsed).

At each iteration the probability P is dependant on two variables, ΔE (the change in conflict, measured by the difference in cost between the new and current states) and T (the current temperature), and is defined as:

$$P = e^{-\Delta E/T}.$$

T is assigned a relatively high initial value; and decreased in steps throughout the running of the algorithm. At high temperatures, poor displacements (large negative ΔE) will often be accepted. At low temperatures, poor displacements will tend to be rejected (although displacements resulting in small negative ΔE might still sometimes be accepted). The acceptance of some poor displacements is permitted so as to allow an escape from locally optimal solutions. In practice, the probability P, is usually tested against a random number R ($0 \leq R \leq 1$). A value of $R < P$ results in the new state being accepted. For example, if $P = 1/3$, then on average, every third worse new state to be accepted. The initial value of T and the rate by which it decreases is governed by what is called the annealing or cooling schedule. Generally, the higher the initial temperature and the slower the rate of change, the better the result (in cost reduction terms). However, the processing overheads associated with the algorithm will increase as the rate of change in T becomes more gradual.

Finding a minimum cost configuration, D_{min}, by SA is statistically guaranteed, provided that the temperature reductions are small enough, and that for each temperature the number of configurations tested is large enough (Zoraster 1997). However, most practical applications settle for near optimal solutions, and make corresponding compromises in the annealing schedule; a suitable schedule is usually decided upon after some preliminary experimentation.

3.3.4 Cost Function

The viability of any iterative improvement algorithm depends heavily on it having an efficient cost function, the purpose of which is to determine for any given element of the search space (i.e. any map realisation) a value that represents the relative quality of that element. The measure of the quality is based on minimising the total number of conflicts within a particular element (i.e. the fewer the conflicts, the better the solution). The cost

function used here, C, is called repeatedly and works by calculating and summing the costs associated with objects $m_i \in M$.

When invoked initially, C must calculate the cost associated with every object $m_i \in M$. A record of these costs is maintained for future reference, meaning that, in any further call, C has to consider only objects with costs affected by the most recent displacement. Calculating the cost associated with a polygonal object, m_i, involves identifying all other polygonal objects lying within a distance, d_{min1}, and all linear objects lying within a distance, d_{min2}. Identifying these conflicting objects quickly requires the use of a spatial index of some kind. In order to satisfy this requirement, C makes extensive use of a triangulation based conflict detection procedure as described in Ware and Jones (1996).

3.4 Results

Initial experiments demonstrate that both gradient descent and SA approaches are effective in reducing conflict while limiting the number of realisations examined. When compared against each other, the SA approach is clearly superior with regard to the degree of conflict reduction achieved.

The experiments make use of five hand generated data sets and IGN-France BDTopo data (1:25,000). The BDTopo data consists of 321 polygonal objects contained within 16 regions (Figure 3.9). For the BDTopo experiment, the minimum separating distance tolerances used assume a visual perception threshold of 0.15mm and a map scale reduction to 1:50,000. The tolerance values (d_{min1} and d_{min2}), displacement value (d) and initial conflict values are shown in Table 3.2. In the experiments, a Type-1 conflict is deemed less serious than a Type-2 conflict; the cost values, c_1 and c_2, are set so as to reflect this fact. The experiments made use of a C implementation of the SA algorithm (compiled with –O3 option) running under UNIX on a Sun Enterprise 2 model 2200 (2x200MHz Ultrasparc). Table 3.1 details the annealing schedule parameters that were used in the experiments.

It can be seen from the results given in Table 3.3; and illustrated in Figure 3.9 and 3.10. The SA approach reduces the amount of graphic conflict by up to 90%. It achieves this

while at the same time limiting the number of realisations having to be generated and evaluated (approximately 300,000 out of a possible 29^{321}).

τ	λ	ω	ξ	ψ
3	0.1	100	30	50

τ Initial temp, λ Reduction factor, ω Placed factor, ξ Successful placed factor, ψ Max temp stages.
Table 3.1: Annealing schedule used in SA.

Data set	d_{min1}	d_{min2}	d	Type-1 conflict	Type-2 conflict
1	2.0	1.5	2.0	10	4
2	3.0	3.0	3.0	6	4
3	2.0	1.5	2.0	12	5
4	3.0	3.0	3.0	12	7
5	2.0	1.5	3.0	26	1
BDTopo	7.5	7.5	7.5	236	36

Table 3.2: Tolerance values, displacement values and initial conflict values.

Data set	Type-1 conflict average	s.d.	Type-2 conflict average	s.d.	Realisations tested average	s.d.	Time taken (s) average	s.d.
1	0.0	(0.0)	0.0	(0.0)	1	0.0	(0.0)	0.0
2	0.0	(0.0)	0.0	(0.0)	2	0.0	(0.0)	0.0
3	0.0	(0.0)	0.0	(0.0)	3	0.0	(0.0)	0.0
4	0.0	(0.0)	0.0	(0.0)	4	0.0	(0.0)	0.0
5	0.4	(0.8)	0.0	(0.0)	5	0.4	(0.8)	0.0
BDTopo	26.6	(2.9)	0.0	(0.0)	BDTopo	26.6	(2.9)	0.0

Table 3.3: Results obtained using SA algorithm.

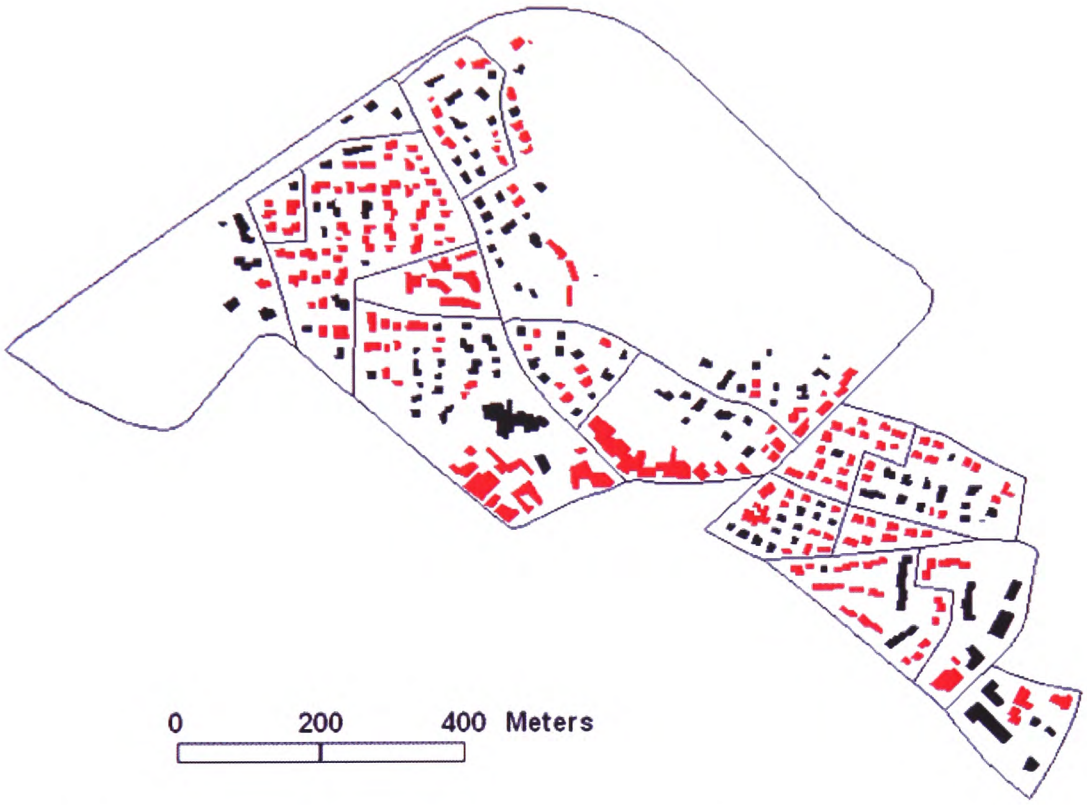


Figure 3.9: IGN BDTopo data before application of SA algorithm. Buildings coloured red are involved in some kind of graphic conflict.

Figure 3.10: IGN BDTopo data after application of SA algorithm. Buildings coloured red are involved in some kind of graphic conflict.

3.5 Summary

This chapter overviews previous work carried out at the University of Glamorgan by Ware and Jones (1998). This research focused upon using a trial position strategy for displacing areal objects such as buildings. Due to the sheer number of potential map realisations, SA is used to limit the number of alternative displays that are generated and evaluated. In order for the generalisation process to succeed, the system incorporates an effective cost function the purpose of which is to ascertain the overall quality of each realisation. The system was implemented in the C programming language and tested on various data sets, the results of which have been discussed.

Chapter 4

Execution Time Improvements

This chapter describes two techniques that have been developed in order to speed-up the SA generalisation tool. Execution times for the original implementation are too slow for it to be of practical use on large data sets and in applications requiring on-the-fly generalisation (e.g. location based services, web mapping and in-car navigation systems etc). The theory and methodology of both techniques are described, and their effectiveness in reducing execution times is demonstrated.

4.1 Accelerating SA

Since its introduction in the early 1980s, several authors have addressed the issue of accelerating the original SA technique, and promising results have been achieved (Varanelli 1996, Gelfand *et. al* 1985, Zoraster 1997). Research has focused on determining the optimum cooling (or annealing) schedules for a series of different problems (Hajek 1988, Kirkpatrick *et. al* 1983). In particular, the work of Varanelli (1996) has attracted much interest. In his paper, Varanelli present two methods for improving execution times; the first involves adopting a two stage simulated annealing schedule, the second involves the use of parallelisation techniques. Many of his experiments rely on applying these modifications to combinatorial problems such as Very Large Scale Integration (VLSI) network partitioning and the well-known travelling salesperson problem.

The two acceleration techniques that are incorporated into the SA generalisation software are: (i) Map partitioning – that is, dividing the map space into autonomous regions (known as segments), each of which is processed separately; and (ii) Two-stage annealing - borrowing principles from Varanelli (1996) and Zoraster (1997).

4.2 Initial Results

For the sake of valid comparison with results reported later, the experiments published in Chapter 3 have been repeated using the parameters outlined in Table 4.1, but this time using a C++ version of the SA software run under a Windows XP operating system on a Pentium 4 2.5GHz machine with 1 Gigabyte of RAM. In order to provide an effective means for evaluation, all future experiments were carried out under these conditions. Furthermore, the evaluation made use of OS™ MasterMap® data sets of which two areas from the Isle of Wight have been specifically identified for use in the experiments (see Figure 4.1 and Figure 4.2). The result of running the C++ version of the SA software on both data sets is shown in Table 4.2.

Figure 4.1: OS™ MasterMap® Data set 1 with OS™ OSCAR® road centrelines.

Figure 4.2: OS™ MasterMap® Data set 2 with OS™ OSCAR® road centrelines.

A slight change has also been made to the algorithm's "cooling" schedule. Instead of T being decreased linearly (i.e. $T_{new} = T_{old} - \lambda\tau$), it is reduced geometrically (as suggest by Zoraster, 1997), such that $T_{new} = \lambda T_{old}$. This change is reflected in the results presented in the remainder of the thesis.

Stage	τ	λ	ω	ξ	ψ
1	500	0.95	10	5	50
			Cut off temperature = 0.1 Random seed = -6		

Scheduling parameters used in SA.
τ Initial temp, λ Reduction factor, ω Placed factor, ξ Successful placed factor, ψ Max temp stages.

Type-1 conflict	Type-2 conflict
50	500

Cost settings for various types of conflict.

Type-1 tol	Type-2 tol	D_{disp}
1.0	10.0	10.0

Building tolerance values in metres (*provided by OS ™*).

Table 4.1: Initial parameters used in SA.

	Initial			Final Results				
Data set	Type-1 conflict	Type-2 conflict	Conflict value	Type-1 conflict	Type-2 conflict	Conflict value	Total no. of tests	Execution time (secs)
1	243	66	146400	13	2	45394	1562928	131.363
2	80	12	40600	12	0	11958	196114	12.571

Table 4.2: Initial results.

Using the default annealing schedule in Table 4.1, SA manages to resolve all but 15 graphic conflicts for data set 1, taking approximately 131.36s to execute and leaving a final conflict value of 45,394. Dataset 2 is processed in 12.571s resulting in a final conflict value of 11,958.

4.3 Data Partitioning

The first technique for speeding up execution times involves dividing the map into autonomous regions (i.e. such that there is no possibility of objects in one region coming into conflict with objects in any other region). The reasoning here is that by segmenting the data, the annealing process (specifically τ and the rate of cooling) can respond to meet more appropriately the specific demands of each region.

One problem to be solved here is finding a technique for dividing up the map. In some instances it may be possible to make use of naturally occurring regions, such as those formed by a road network or administrative boundaries. Other situations will require analysis of the distribution of objects in order to find groupings of objects that sustain no influence from objects external to their group. In this work, a road network (OS™ OSCAR®) is used to divide the map into independent regions (see Figure 4.3 and Figure 4.4); the technique was used previously by Ruas (1998). Currently the segmentation process is undertaken manually using editing tools in ESRI's™ ARCMAP™ software. The development of a function to automate this task was not investigated in this research.

Figure 4.3: Data set 1 partitioned into 14 regions using OS™ OSCAR® road centrelines.

Figure 4.4: Data set 2 partitioned into 9 regions using OS™ OSCAR® road centrelines.

4.4 Optimisation Approaches

4.4.1 Starting Temperature

One of the key difficulties in evaluating SA is finding the optimum starting temperature for a particular problem. The starting temperature τ must be high enough so as to allow a move to almost any neighbourhood state (in this case an alternative map realisation), otherwise it is likely that the final solution will either be the same as or close to the starting solution. However, if the temperature starts at too high a value then the search becomes randomised and remains like this until the temperature is cooled sufficiently, at which point it will begin acting again as SA. Rayward-Smith (1996) and Dowsland (1995) propose similar methods to try and find the correct starting temperature. Their proposals start the annealing process with a very high temperature and cool it rapidly so that the majority of worst solutions are accepted. From this a more acceptable starting temperature can be found, and which can then be cooled at a much slower rate. The technique described by Zoraster (1997) is adopted in this research (see next section).

4.4.2 Automatic Setting of Starting Temperature

A common problem uncovered in this research is finding the correct starting temperature for each of the partitioned map regions manually. Finding the optimum set of annealing values for each region is a tedious and time consuming process and one which renders itself totally unpractical for batch-mode processing. In-fact the time taken to try and find an optimum set of values (using a manual strategy) for a particular map region would significantly be beyond the execution times achieved in the initial experiments. A solution to this problem is to partly automate the annealing schedule setting process, or to be more precise, to automate the setting of the starting temperature.

Previously the value τ was supplied as a user defined input parameter (a suitable value being arrived at empirically). A more flexible approach is now adopted, in which τ is generated automatically. For the first φ iterations, the algorithm accepts non-improving moves with probability, $1/3$. At this point the mean value $\mathbf{m}(\Delta E)$ for non-improving moves encountered (during the φ iterations) is calculated. τ is then computed so that P is $1/3$ for the average value of $\mathbf{m}(\Delta E)$:

$$\tau = \mathbf{m}(\Delta E)/(\ln 3)$$

Experiments show that $\varphi = 500$ works well for all data sets.

In Table 4.1 the user defined starting temperature τ was set to a fixed value of 500. In order to provide a fair comparison against the partitioned results (below), the experiments in §4.2 are repeated using an automated starting temperature and applied globally to data set 1 and data set 2. A summary of the updated results can be found in Table 4.3.

		Initial			*Final Results*				
Data set	τ *(auto)*	*Type-1 conflict*	*Type-2 conflict*	*Conflict value*	*Type-1 conflict*	*Type-2 conflict*	*Conflict value*	*Total no. of tests*	*Execution time (secs)*
1	478.7	243	66	146400	13	2	45642	1561461	131.144
2	84.5	80	12	40600	12	0	12032	195011	12.432

Table 4.3: Updated results for data set 1 and data set 2 with automated initial starting temperatures.

Partitioned Results

The following tables provide evidence of benefits gained by applying SA to the segmented versions of data sets 1 and 2. Execution times for data set 1 are reduced to 114.267s (previously 131.144s), while data set 2 is processed in 9.096s (previously 12.432s). A decrease in execution times of ~13% for data set 1 and ~28% for data set 2.

		Initial			Final Results				
Region	τ (auto)	Type-1 conflict	Type-2 conflict	Conflict value	Type-1 conflict	Type-2 conflict	Conflict value	Total no. of tests	Execution time (secs)
1	103.0	50	2	25100	4	0	3556	154810	10.963
2	72.67	17	0	8500	1	0	1113	79300	6.229
3	56.31	6	4	3200	1	0	714	15400	13.42
4	76.54	11	0	5500	0	0	495	45094	2.904
5	83.37	2	0	1000	0	0	108	24250	1.592
6	152.2	6	0	3000	0	0	140	8040	0.591
7	105.9	19	0	9500	0	0	4778	152070	11.227
8	79.73	33	6	16800	0	0	6730	211200	16.684
9	78.38	22	2	11100	1	0	1208	45120	3.355
10	91.58	6	2	3100	0	0	327	34960	2.724
11	153.5	75	2	37600	3	0	5399	181250	20.771
12	123.4	5	0	2500	0	0	196	20370	1.763
13	151.7	14	0	7000	1	0	2794	52655	5.138
14	86.31	25	0	12500	1	0	3003	202160	16.906
Total		**243**	**66**	**146400**	**12**	**0**	**30561**	**1226679**	**114.267**

Table 4.4: Updated results for partitioned data set 1 with automated initial starting temperatures.

		Initial			Final Results				
Region	τ (auto)	Type-1 conflict	Type-2 conflict	Conflict value	Type-1 conflict	Type-2 conflict	Conflict value	Total no. of tests	Execution time (secs)
1	63.2	5	2	2600	2	0	1174	8970	0.51
2	113.17	10	0	5000	2	0	1797	22080	1.583
3	268.65	3	0	1500	2	0	1600	3060	0.221
4	89.9	11	2	5600	2	0	1413	9690	0.531
5	120.74	2	0	1000	0	0	81	3660	0.201
6	219.12	5	2	2600	2	0	1651	8800	0.441
7	80.53	17	4	8700	1	0	1111	44820	2.755
8	68.17	6	0	3000	1	0	1295	9000	0.52
9	56.02	21	2	10600	2	0	2688	39480	2.334
Total		**80**	**12**	**40600**	**14**	**0**	**12810**	**149560**	**9.096**

Table 4.5: Updated results for partitioned data set 2 with automated initial starting temperatures.

4.5 Two-stage Annealing

Some authors suggest that optimisation can be made more efficient by adopting a two-stage approach (e.g. Varanelli and Cohoon 1993). In two-stage SA, a faster heuristic algorithm is used to replace the SA actions occurring at higher temperatures in an attempt to advance on the initial solution, while a slower heuristic algorithm is used to replace the SA actions occurring at lower temperatures.

The approach adopted in the implementation presented here differs slightly in that the faster heuristic algorithm is replaced by SA in conjunction with an annealing schedule that involves the first stage starting with a high temperature followed by rapid cooling. The second stage again makes use of SA but this time starting with a much lower initial temperature followed by a slower rate of cooling (see Figure 4.5). For the first stage, the initial starting temperature is set automatically. For the second stage, experiments showed that more consistent results were obtained by reverting to a fixed value of τ. (see Table 4.6). Experimental results of applying two-stage annealing to partitioned regions in data sets 1 and 2 are presented in Tables 4.7 and 4.8. Execution times are reduced from 114.267 to 79.194s for data set 1, while data set 2 is processed in 7.525s (previously 9.096); a decrease in execution times of ~40% when compared to original timings.

Stage	τ	λ	ω	ξ	ψ
1	auto	0.2	10	5	50
2	50	0.9	20	10	50
			Cut off temperature = 0.1		
			Random seed = -6		

Table 4.6: Annealing parameters used in two-stage SA.
τ Initial temp, λ Reduction factor, ω Placed factor, ξ Successful placed factor, ψ Max temp stages.

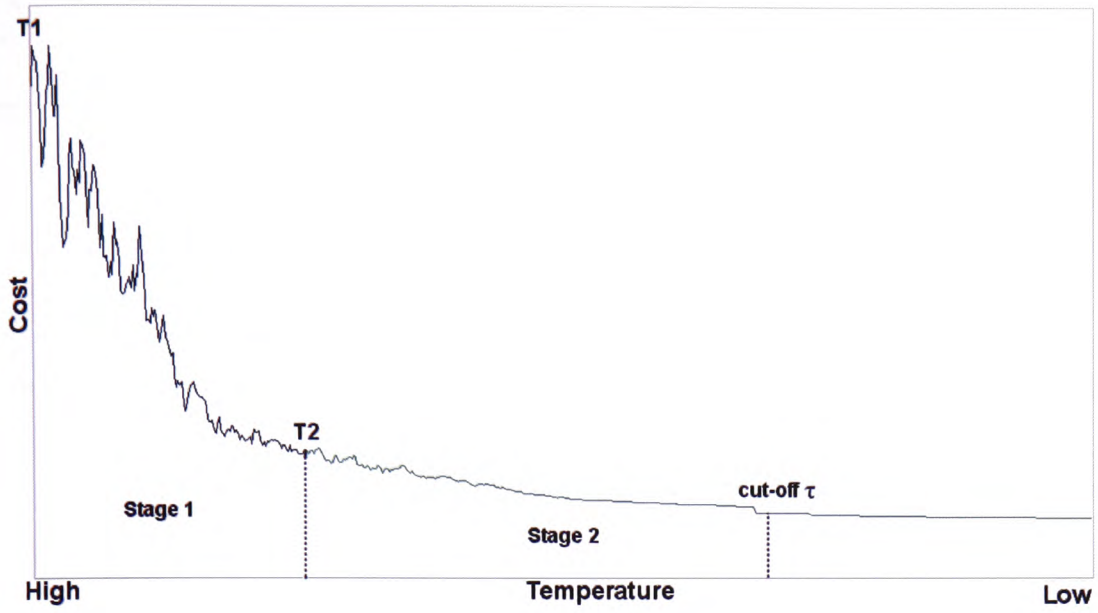


Figure 4.5: Plot of a typical two-stage annealing schedule.
T1 – Starting temperature for stage 1; T2 – Starting temperature for stage 2; cut-off τ (temperature)
A faster heuristic algorithm is used to replace the SA actions occurring at higher temperatures (stage 1) in an attempt to advance on the initial solution, while a slower heuristic algorithm (stage 2) is used to replace the SA actions occurring at lower temperatures.

Two-stage Partitioned Results

		Initial			*Final Results*				
Region	τ	*Type-1 conflict*	*Type-2 conflict*	*Conflict value*	*Type-1 conflict*	*Type-2 conflict*	*Conflict value*	*Total no. of tests*	*Execution time (secs)*
1	103.0	50	2	25100	4	0	3462	104607	7.61
2	72.67	17	0	8500	1	0	1076	35336	2.64
3	56.31	6	4	3200	0	0	104	13200	1.193
4	76.54	11	0	5500	0	0	185	37470	2.454
5	83.37	2	0	1000	0	0	121	30211	1.267
6	152.2	6	0	3000	0	0	60	7560	0.55
7	105.9	19	0	9500	1	0	1120	38271	8.52
8	79.73	33	6	16800	0	0	3765	145600	11.947
9	78.38	22	2	11100	0	0	1289	35250	2.654
10	91.58	6	2	3100	1	0	747	45050	3.525
11	153.5	75	2	37600	4	0	10098	193980	18.527
12	123.4	5	0	2500	0	0	72	7401	0.911
13	151.7	14	0	7000	0	0	1385	13545	5.268
14	86.31	25	0	12500	1	0	4303	145920	12.128
Total		**243**	**66**	**146400**	**12**	**0**	**27787**	**853401**	**79.194**

Table 4.7: Updated results for two-stage SA.

Region	τ	Initial			Final Results				
		Type-1 conflict	Type-2 conflict	Conflict value	Type-1 conflict	Type-2 conflict	Conflict value	Total no. of tests	Execution time (secs)
1	63.2	5	2	2600	2	0	1132	4810	0.271
2	113.17	10	0	5000	1	0	1380	22421	1.072
3	268.65	3	0	1500	2	0	1035	4680	0.301
4	89.9	11	2	5600	2	0	1098	9350	0.621
5	120.74	2	0	1000	0	0	79	2721	0.132
6	219.12	5	2	2600	2	0	1118	6160	0.26
7	80.53	17	4	8700	1	0	834	42120	2.743
8	68.17	6	0	3000	1	0	1256	9300	0.572
9	56.02	21	2	10600	1	0	1275	23520	1.553
Total		**80**	**12**	**40600**	**12**	**0**	**9207**	**125082**	**7.525**

Table 4.8: Updated results for two-stage SA.

4.6 Summary

This chapter presents techniques for improving the execution times of the SA implementation. A method is used that partitions each data set into autonomous regions so that the annealing process can respond to meet more appropriately the specific demands of each region. Determining the optimum initial starting temperature using manual techniques (empirically) for each region is difficult and impractical. As a result this chapter has investigated an automated approach to assist in this task. A final attempt at reducing execution times further has been to investigate a technique based on two-stage annealing, which is combined with the data partitioning strategy.

Chapter 5

Incorporating Additional Operators

This chapter tackles the problem of reducing graphic conflict further by incorporating additional operators into the SA solution. It presents detail on how this is accomplished and describes a series of experiments that demonstrate the usefulness of the approach.

5.1 Limitations of Displacement

By examining results in previous chapters, it can be seen that the sole use of the displacement operator does not guarantee the removal of all graphic conflict. This becomes most apparent in situations where there is very little free map space into which map objects can move. This is illustrated in Figure 5.1. For example the best result obtained during the initial experiments for data set 1 was a final Type-1 & Type-2 conflict cost of 45,394 with 15 conflicts left unresolved. Similar results can be found for the data set 2 (see Table 4.2, Chapter 4).

It is unacceptable from a cartographic perspective to simply remove objects that cannot be resolved using the displacement operator. Therefore a solution is presented in which additional operators are incorporated into the SA solution. The work presented here makes use of three additional operators: object size reduction, object size enlargement and object deletion.

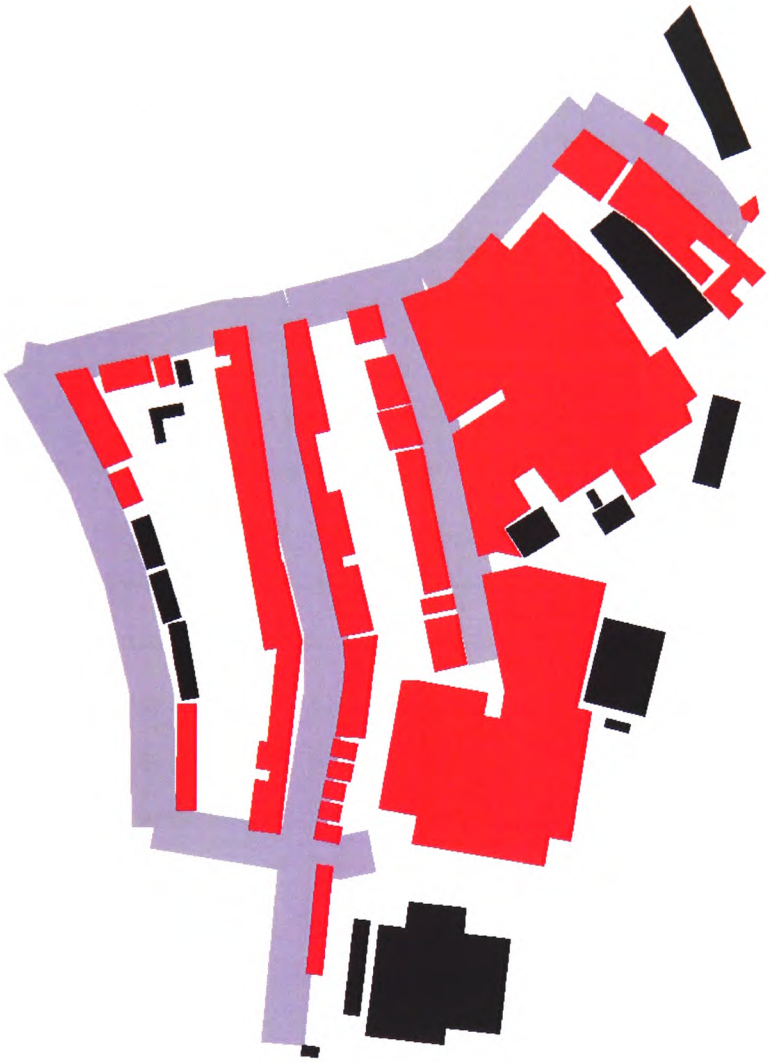
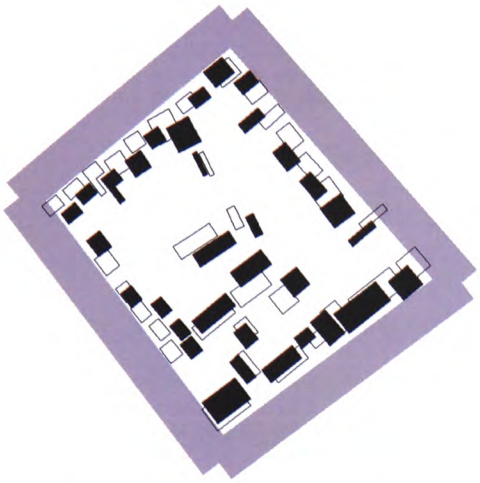


Figure 5.1: Consequences of scale-reduction in which roads are symbolised, resulting in conflict between buildings. In this example, there is very little free space into which buildings can be displaced.

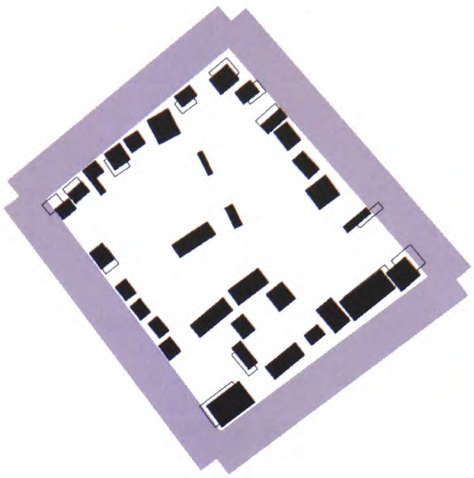
5.2 Displacement Cost

Before continuing with the discussion on additional operators, it is first necessary to highlight some changes that have been made in relation to the displacement operator and the existing cost function. At present, map objects are displaced during the course of annealing with the overall aim being to reduce graphic conflict. However it is observed that many of the displacements prove unnecessary (i.e. they do not lead ultimately to a reduction in graphic conflict) and occur only as a consequence of the algorithm's occasional acceptance of neutral and negative displacement. The result is a final display containing objects displaced from their original location without benefit (see Figure 5.2). In order to minimize displacement of this type an object displacement cost is introduced. That is, if an object exists in a displaced state then a cost δc_3 is incurred, where δ represents the magnitude of the displacement.

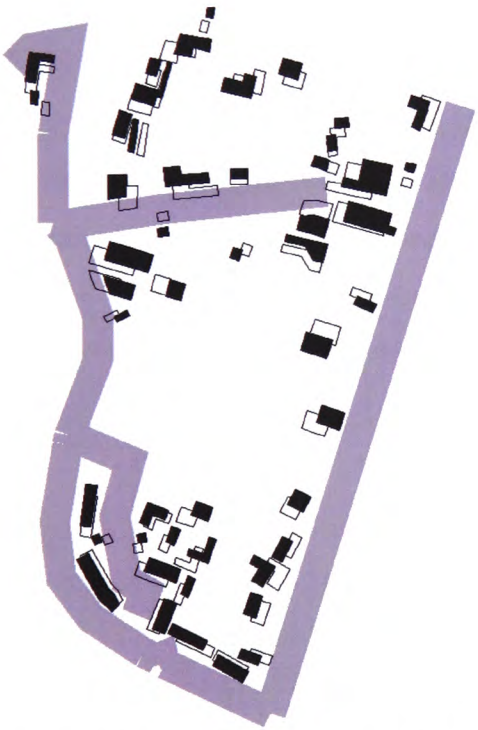
Therefore the cost associated with a particular object is now a combination of its graphic conflict along with its displacement cost. For example, consider an object m_i that currently exists in a displaced state (cost = δc_3) and lies in conflict with two other polygonal objects (cost = $2c_1$) and one linear object (cost = c_2). Its associated cost would equal ($2c_1 + c_2 + \delta c_3$). Assigning appropriate values to c_1, c_2, and c_3 (i.e. $0 < c_3 < c_2, < c_3$) creates an incentive for displaced objects to return, during the course of annealing, as near to their original location as is possible without a resulting increase in conflict. Displays produced with and without consideration to displacement cost are shown in Figure 5.2.



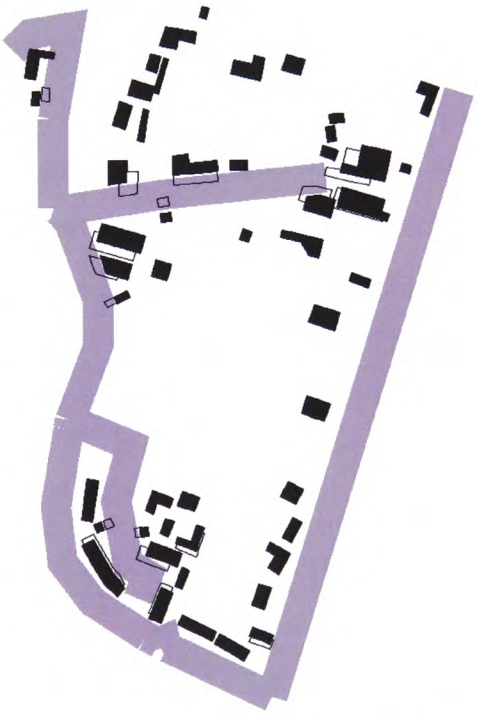
Example 1 (a) – Without displacement cost.



Example 1 (b) – With displacement cost.



Example 2 (a) – Without displacement cost.



Example 2 (b) – With displacement cost.

Figure 5.2: Examples of incorporating displacement cost.
Original location of building features are displayed as an outline, while displaced features are shown as solid.

5.3 Incorporating Additional Operators

Three additional operators are now incorporated into the existing SA technique in an attempt to reduce graphic conflict further. These are:

- Object size exaggeration.
- Object deletion.
- Object size reduction.

Object size exaggeration is required in situations where an object of particular importance becomes too small to be viewed adequately at the target scale. In situations where there is not enough free space for all objects to be displayed some non-important objects are deleted. An alternative option to deletion would be to reduce the size of relatively large objects.

In implementation terms, introducing size enlargement, size reduction and deletion capabilities is straightforward; these additional modified states of an object are treated as additional trial positions for that object (see Figure 5.3). Size reduction and size enlargement are achieved by applying a suitable scaling factor (s_r and s_e) to the object (see Figure 5.4), while deletion is accommodated by means of a simple Boolean flag (i.e. an object is either deleted or not deleted).

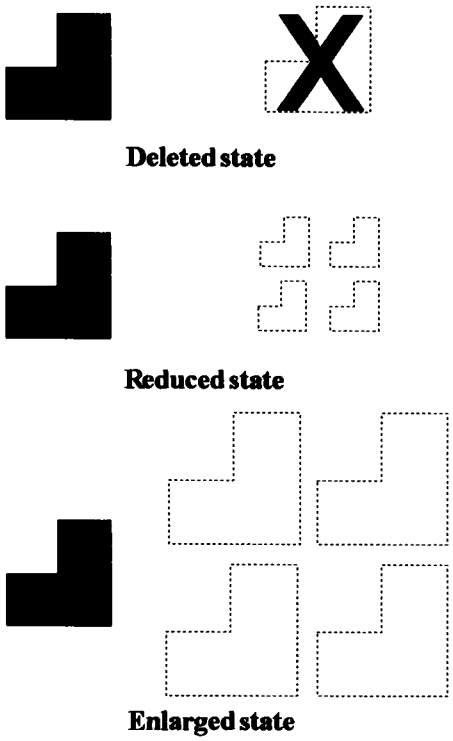


Figure 5.3: Additional object states.

```
void ScalePoints(int *vert_list, int number_of_verts, double s, dx, dy)
{
 int i;
 for (i = 0; i < number_of_verts; i++)
 {
  point_array[vert_list[i]].x_value = (point_array[vert_list[i]].x_value - dx) * s + dx;
  point_array[vert_list[i]].y_value = (point_array[vert_list[i]].y_value - dy) * s + dy;
 }
 Return;
}
```

Figure 5.4: Scaling function in C++. s – scaling factor; dx, dy – centre of gravity of vertices.

5.3.1 Formalising Additional Object States

To expand upon the previous discussion, it is now necessary to consider a map display D made up of fixed linear objects, F, and n modifiable detached polygonal objects, M. Each modifiable object, $m_i \in M$, has k possible states providing us with k^n possible configurations of D. At any given time a particular object m_i exists in one of its k possible states; its $(k\text{-}1)$ modified states arise as a result of either displacing the object, reducing the size of the object, increasing the size of the object or deleting the object.

In the current implementation, the possible object states are:

- *Unmodified State.* Each object m_i has a single unmodified state.
- *Deleted State.* An object m_i has a single deleted state trial position; this trial position represents the situation where the object has been removed from the display.
- *Displaced States.* Each object m_i has q displaced state trial positions, which are distributed evenly about the object.
- *Reduced States.* Each object m_i has $q+1$ reduced state trial positions; these trial positions result from applying a scaling factor s_r ($0<s_r<1$) to the unmodified state and displaced states of m_i.
- *Enlarged States.* Each object m_i has $q+1$ enlarged state trial position; these trial positions result from applying a scaling factor s_e ($s_e \geq 1$) to the unmodified state and displaced states of m_i.

The object reduction value, s_r, is fixed for all objects (e.g. 0.94). However this value can be changed at run-time according to the amount of graphic reduction required. The object enlargement value, s_e varies for each object and is dependant on object display area. For an object with display area less than a minimum area tolerance a_{min}, s_e is set so as to increase object display area to a_{min}. Objects with display area greater than or equal to a_{min} have s_e set to 1 (i.e. its application has no effect).

5.3.2 Updated Evaluation of Map Display

For a particular configuration D_j, an object m_i has an associated cost. This cost is a measure of both the graphic conflict in which the object is involved and the extent to which the object is modified. There are now three categories of graphic conflict. The first two are the Type-1 and Type-2 described previously (Chapter 3). The third, Type-3, occurs when the area of m_i (in viewing co-ordinates) is less than a_{min} (i.e. m_i is too small for it to be seen clearly). a_{min} is compared against each object in turn and for those objects whose minimum physical area falls below a user defined value (specified by the OS™) use is made of the enlargement operator which automatically increases the size of an object to a_{min}. An occurrence of Type-3 conflict carries a cost c_3.

If an object exists in a modified state then a modification cost is incurred. Displacing an object carries a cost δc_4, where δ represents the magnitude of displacement from the original position. Reducing an object carries a cost c_5, with c_5 being proportional to the scale of reduction, relative to the original size. Enlarging an object carries a cost, c_6, which is proportional to the scale of enlargement, relative to the original size. Deleting an object carries a cost c_7.

The costs associated with graphic conflict and object modification are combined to give the overall cost associated with an object. For example, consider an object m_i that has been reduced in size and lies in conflict with two other polygonal objects and one linear object. Its associated cost would equal $(2c_1 + c_2 + c_5)$. As before, the total cost $\mathbf{C}(D_j)$ associated with a realisation D_j is found by summing the costs associated with each object m_i. The goal is to find a minimum cost configuration D_{min}.

5.3.3 Operator Precedence Using Weights

It is important to make sure that conflict costs (c_1, c_2 and c_3) and object modification costs (c_4, c_5, c_6 and c_7) are set appropriately; it is these costs that govern the likelihood of any given object/trial position pairing being accepted should they be chosen during the annealing process. As such, the cost values must be set so as to accommodate any orders of precedence that might exist between the various operators and conflict types. For example, consider an application that simply requires the removal of all graphic conflict

(i.e. reducing graphic conflict is the stated primary goal, and minimising object modification is an implied secondary goal). This might be achieved by assigning a relatively high value to each of the conflict costs (e.g. c_1=100, c_2=100 and c_3=100) and a relatively low value to each of the object modification costs (e.g. c_4= 5, c_5=5, c_6=5 and c_7=5). It could be the case, however, that the modification operators have an order of precedence in which object displacement is preferred to object reduction, which in turn is preferred to object deletion; the relevant cost values are changed accordingly (c_4= 5, c_5=10 and c_7=15). In the evaluation, the effects of applying various priorities to different operators are explored.

5.4 Building Importance Factor

When generalising a map it is important to consider the relative importance of objects; important objects should be less prone to modification than unimportant objects. Consider a tourist map in which an object m_m, representing a museum, lies in conflict with an object m_h, representing an ordinary house. In this context, m_m can be regarded as being more important than m_h, and hence should be less susceptible to generalisation. In this situation there are a number of alternative conflict resolution strategies that could be employed. For example, conflict resolution could initially involve displacement and reduction of m_h only; if this failed, m_h could be deleted. An alternative strategy might again involve the initial displacement and reduction of m_h only. If this failed then the next step would be to displace and reduce m_h and m_m in combination; continued failure at this stage would result in the deletion of m_h. Relative object importance is incorporated into the SA procedure by assigning a cost weighting w_i to each object m_i. Whenever an object, m_i, and trial position, k_j, pairing are chosen during the annealing process, the cost associated with k_j is multiplied by w_i to give a modified cost. A particular cost weighting value, w_i, will be based on one or more of the attributes of m_i. In the experiments to date, and in the absence of any other measure of importance, an object's importance, and hence its cost weighting value, is assumed to be proportional to object area, with large objects deemed more important than small objects. The larger the size of the building the greater its importance factor (see Figure 5.5).

If necessary alternative strategies can be applied for determining an objects importance by extracting information from a building attribute table (e.g. type of building). Using this

additional information, features that are similar in some aspect (e.g. house, factory) may be grouped together. However, at the time of writing, this information was not available with the OS™ MasterMap® data sets made available for the project.

Figure 5.5: Building importance factor in SA.
In this example (a), smaller, less-important buildings are deleted (to free-up map space) to allow the displacement of larger, more important buildings to take place (b).

5.5 Evaluation

The purpose of this evaluation is to demonstrate the efficiency (in relation to the removal of graphic conflict) of including additional operators into the SA procedure. In the experiments the aim is to eliminate as much graphic conflict as possible using a combination of displacement, reduction, enlargement and deletion operators.

Experiment 1 (Displacement and Reduction)

The first experiment makes use of displacement in conjunction with the enlargement and reduction operators. Traditionally, cartographers at the OS™ would normally avoid reducing objects (especially smaller buildings) in an attempt to remove graphic conflict. Instead their approach relies on conventional clipping methods (e.g. if a building is in conflict with a road, the area of the building which overlaps the road is removed). This has the disadvantage that certain map objects that are irregular in shape will become geometrically distorted. The approach presented in this work preserves feature shape. However, care has to be taken and in order to satisfy the OS™ generalisation specifications it is necessary to ensure that only large objects are reduced in size and then only by relatively small amounts. Small objects are thus excluded from any reduction transformations. The enlargement operator is applied to smaller objects whose area falls below a_{min} to increase the size of an object to a value specified by the minimum building area (MBA) (see Table 5.2).

Using the principles of weights as a means of prioritising operators, appropriate values are assigned to displacement, reduction and enlargement operators (see Table 5.1).

Table 5.2 provides tolerance values as suggested to be used when performing generalisation to a target scale of 1:10,000 (provided by the OS™).

Type-1 conflict	Type-2 conflict	Del cost	Red cost	Too small cost
50	500	10	25	1

Table 5.1: Cost settings for various types of conflict
Del cost – Displacement cost; Red cost – Reduction cost;
Too small cost – Building too small cost.

Type-1 tol	Type-2 tol	D_{disp}	Red factor	M.B.A.
1.0	10.0	10.0	0.94	10

Table 5.2: Distance tolerance values specified in metres (*provided by OS*)
Type-1 tolerance; Tyoe-2 tolerance; D – Max displacement
R factor – Building reduction factor; M.B.A – Minimum building area

Results

Results obtained using the displacement and reduction operators are presented in Table 5.4. For sake of a valid comparison the results are compared against the original SA process (i.e. use of the displacement operator only, see Table 5.3). Figures 5.7 and 5.8 show the final output displays of applying various operators to both data sets.

	Initial			*Final Results*			
Data set	Type-1 conflict	Type-2 conflict	Conflict value	Type-1 conflict	Type-2 conflict	Conflict value	Total no of tests
1	243	66	146400	25	2	19590	1206960
2	80	12	40600	21	0	12670	153260

Table 5.3: Results obtained using displacement operator only.

	Initial			*Final Results*			
Data set	Type-1 conflict	Type-2 conflict	Conflict value	Type-1 conflict	Type-2 conflict	Conflict value	Total no of tests
1	243	66	146400	18	2	15950	1209550
2	80	12	40600	13	0	8950	157130

Table 5.4: Results obtained using reduction and displacement operators.

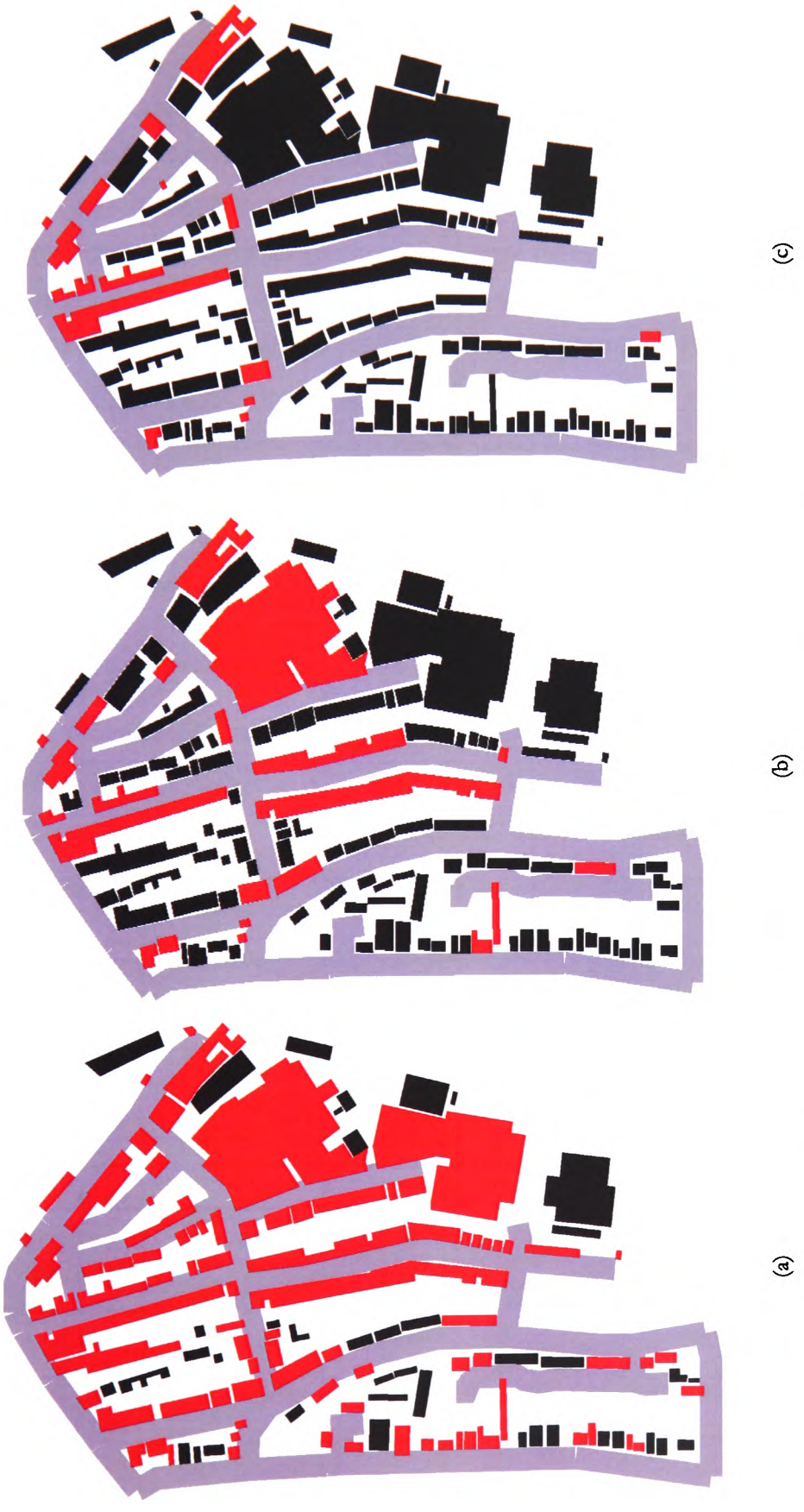
(a)

(b)

(c)

Figure 5.6: Displays produced showing results of additional operators on data set 1. (Conflict highlighted in red). Original map (a); Application of displacement operator (b); Application of displacement and reduction operators (c).

Figure 5.6 displays the output of applying various operators to data set 1. The use of a reduction operator in this case to reduce graphic conflict further only has a partial effect (resolving an additional seven objects) as this is due to the number of small objects that constitute data set 1 (as objects can not be resized below their MBA). Clearly in this situation there is a need for an additional operator to resolve the remaining conflict (e.g. use of the deletion operator). In Figure 5.7 the application of the reduction operator is far more effective in resolving graphic conflict when applied to larger buildings in data set 2.



(a) (b) (c)

Figure 5.7: Displays produced showing results of additional operators on data set 2. (Conflict highlighted in red.) Original map (a); Application of displacement operator (b); Application of displacement and reduction operators (c).

Experiment 2 (Incorporating Deletion)

The aim of the second experiment is to demonstrate the usefulness of using object deletion in conjunction with the displacement and reduction operators. By adopting the building importance strategy as discussed in 5.4, each building is weighted according to its area (i.e. the larger the building the higher the importance factor). The net effect of applying SA is that, hopefully, objects are first displaced away from their original location in an attempt to resolve graphic conflict. Where the resolution of graphic conflict using displacement is not possible, SA then attempts to slightly reduce larger-size buildings to free up sufficient map space. Objects which cannot be displaced or reduced will be deleted depending on their importance factor. In situations where a larger object m_i remains in a state of conflict with a smaller object m_j, SA will be biased towards keeping the larger building and deleting the smaller less-important building. Alternatively, where objects m_i and m_j carry equal importance and are in conflict with one-another, SA makes an arbitrary choice and deletes one of the offending objects (i.e. either m_i or m_j).

Updated Results

Using the updated cost values in Table 5.5 the deletion operator is applied alongside displacement and reduction to previous OS™ MasterMap® data sets. A summary of the results obtained can be found in Table 5.6. Figure 5.8 shows the output display of applying the deletion operator to data set 2.

Type-1 conflict	Type-2 conflict	Dis cost	Red cost	Del cost	Too small cost
50	500	10	25	100	1

Table 5.5: Updated cost settings including deletion operator
Dis cost – Displacement cost; Red cost – Reduction cost; Del cost – Deletion cost; Too small cost – Building too small cost.

	Initial			*Final Results*				
Data set	Type-1 conflict	Type-2 conflict	Conflict value	Type-1 conflict	Type-2 conflict	Objects deleted	Conflict value	Total no of tests
1	243	66	146400	18	0	6	12850	1213452
2	80	12	40600	13	0	10	8350	162320

Table 5.6: Results obtained incorporating the deletion operator.

(a) (b)

Figure 5.8: Displays produced (before and after) incorporating additional operators (displacement, reduction and deletion). Conflict highlighted in red.

5.6 Summary

This chapter has presented the reader with improved strategies for reducing graphic conflict by incorporating additional operators into the original SA process. This is achieved by assigning additional trial positions that correspond to the reduced, enlarged and deleted states of each object. The cost function is also updated to take these additional operators into account.

The assignment of appropriate cost values to each operator allows any orders of precedence that might exist between the various operators and conflict types to be accommodated.

This chapter has also investigated the need for a displacement cost that limits the amount of unnecessary displacement that naturally occurred in the original SA. A series of experiments demonstrates the usefulness of the approach. An in-depth discussion of the results is presented in Chapter 7.

Chapter 6

Refining Simulated Annealing

This chapter provides the reader with a description of work related to advancing SA further by investigating the application of high-order feature alignment and the use of a continuous search space for improved graphic-display output. This chapter also examines the use of additional feature classes.

6.1 Maintaining High-order Feature Alignment

It is undesirable for a map to suffer unnecessary deterioration in quality during the generalisation process. This chapter examines the shortcomings of the current process with a view to increasing the quality of maps that have undergone generalisation. One of the main limitations of the current SA process is that conflict cost is dependent only on the extent to which minimum clearance constraints are violated, and modification cost is calculated as a function of the degree to which each generalisation operator is applied. This can result in an undesired change in alignment of a group of objects that are related in some way and were, prior to the generalisation, aligned in a specific manner. Such a grouping of objects is referred to here as a high-order feature. Experiments show that generalisation without consideration for such features results in a map where individual objects within these features are displaced to such a degree that the feature might no longer be recognisable (see Figure 6.1).

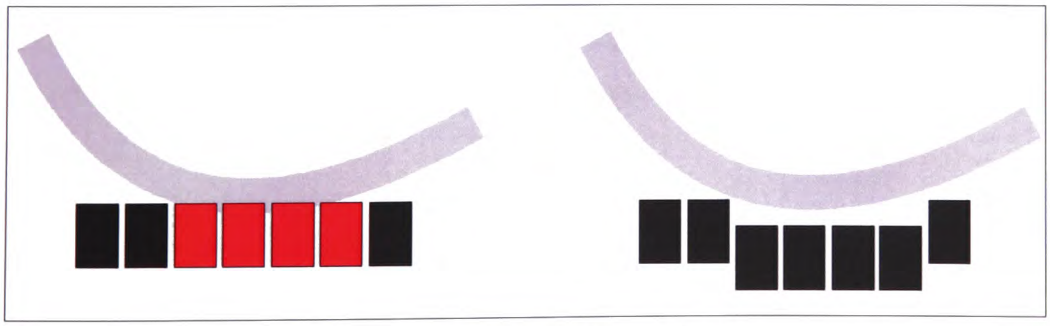


Figure 6.1: Feature misalignment in SA.

High-order features, such as a row of buildings representing a street, are not expected to be explicitly defined within source data sets (as is the case with OS™ MasterMap® data). The first problem to solve therefore is that of grouping together low-order features into higher-order features. The first set of experiments deal with manually generated groupings in order to gain an initial measure on how effectively SA can cope with high order features. A solution to the automated grouping of objects then follows; this solution is based on the assessment of a range of variables, including minimum separating distance between buildings, shape and size of buildings.

In response to the problem of high-order feature disruption, a solution is proposed which involves high-order feature modification (i.e. if a particular object is modified as part of the SA process then all other objects belonging to the same high-order feature undergoes the same modification). This will ensure that spatial relationships between the constituent objects of a high-order feature are maintained (see Figure 6.2).

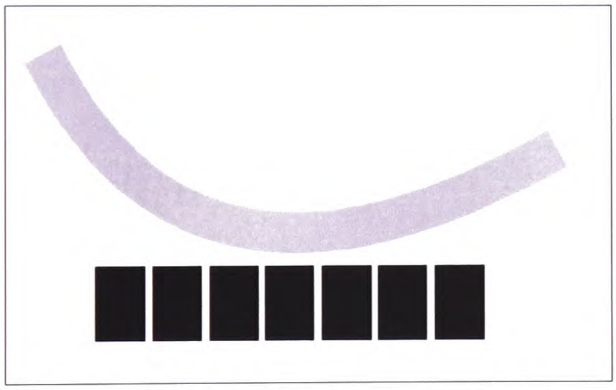


Figure 6.2: Preserving feature alignment in SA.

A disadvantage of the current approach is that conflict may be present between buildings prior to object grouping. This results in conflict being retained within the group itself. A solution to this problem might be to run the SA algorithm locally within each group as a post-process.

6.1.1 What Constitutes a High-order Feature?

A human's ability to group related objects together originates naturally and sometimes without obvious explanation. Humans are capable of instantly recognising shapes, patterns and various similarities between objects, even from an early age with little knowledge and reasoning. When applied to map objects, buildings which look similar can be grouped together. However, while people naturally have the ability to recognise patterns and so group objects, this is not an inherent ability that computers possess. Complex procedures need to be developed and it has been the subject of much interest over the last few decades in various computing fields. In map generalisation, although the subject has received some attention (Christophe and Ruas 2002, Regnauld 1998), little progress has been made.

6.1.2 An Example Using the Reduction Operator

Consider the following problem. A map display D made up of two fixed linear objects, $f_{(i)}$ *and* $f_{(j)}$, and four modifiable detached polygonal objects, $m_{(i)}$, $m_{(j)}$, $m_{(k)}$ *and* $m_{(l)}$ (see Figure 6.3).

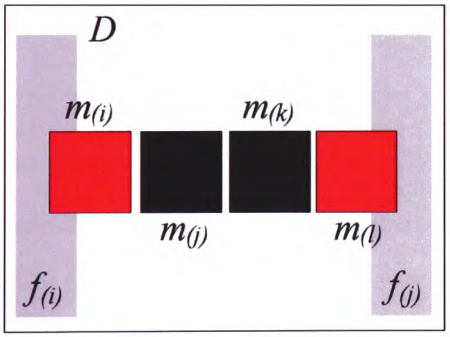


Figure 6.3: Sample data set.

It can be seen from the above figure that two out of the four polygonal objects exist in a state of conflict with a fixed linear object. In the original SA implementation, the algorithm would be unable to resolve this conflict through the use of displacement. Furthermore the use of an additional operator (i.e. reduction), would not result in an appropriate solution since, although the conflict has been resolved by reducing the size of objects, qualitative inconsistencies have been introduced (i.e. the reduction operator appears excessive and has not been applied consistently; see Figure 6.4).

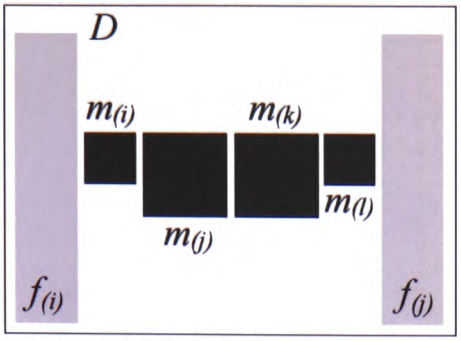


Figure 6.4: Excessive use of the reduction operator.

An alternative approach would be to collectively reduce the size of all four polygonal objects as a group (see Figure 6.5).

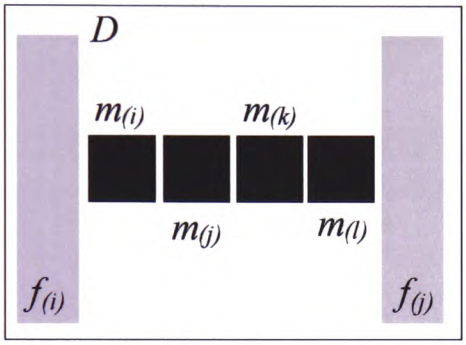


Figure 6.5: Application of process *en masse*.

6.1.3 Object Grouping

6.1.3.1 Manually Generated Groups

The creation of manually generated groups relying on perceptual methods ensures that map objects are organised into their optimal configurations to assess how effectively SA can cope with high-order features. The process begins by recognising objects that qualify for manual grouping as advocated by Wertheimer (1922), and is based on a number of variables which include: shape, size, orientation etc. Suitable groups are then created interactively using ESRI's™ ARCMAP™ software (see Figure 6.6).

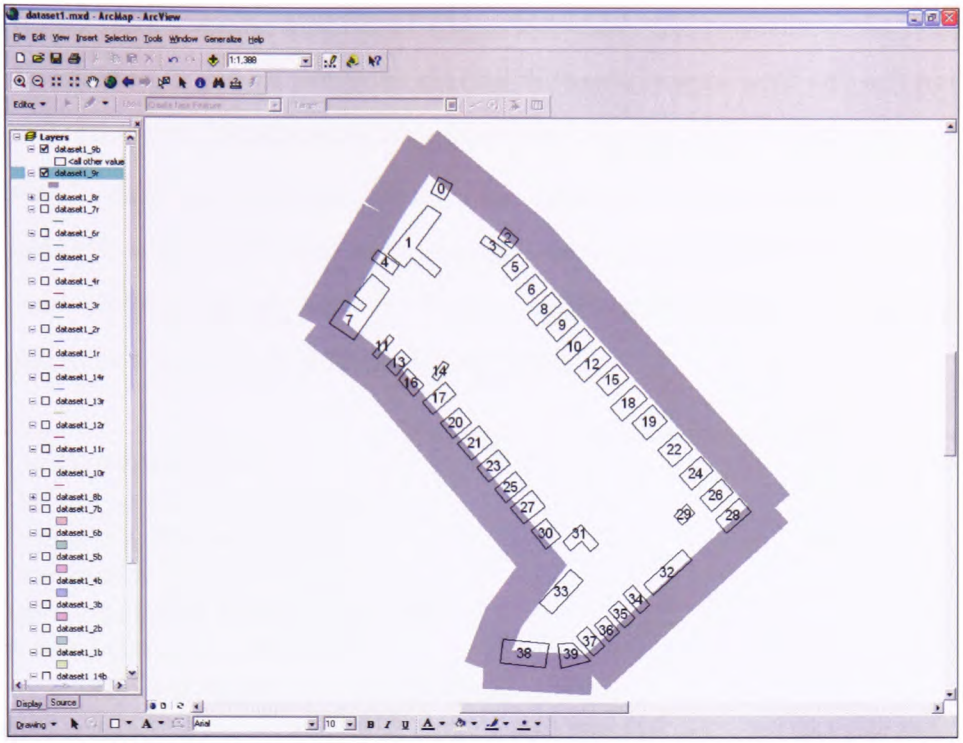


Figure 6.6: Creation of groups within ESRI's™ ARCMAP™ software using ObjectID attribute information.

6.1.3.2 Automated Grouping

In order to provide a fully automated version of this part of the research, a basic polygon object grouping function has been implemented. The function, which is outlined using pseudo-code in Figure 6.7, groups polygons on the basis of separating distance, shape, size and alignment. In short, a pair of objects can be considered as belonging to the same group (i.e. the same high-order feature) if they are:

i) Close enough to each other.
ii) Similar in shape.
iii) Similar in size.
iv) Road aligned.

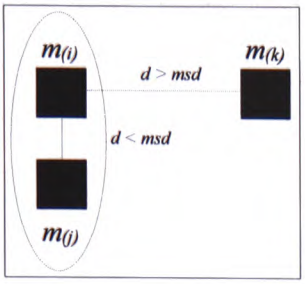
Figure 6.8 illustrates each of these tests. In the implementation, polygon shape is simply calculated as a function of the number of constituent edges. Road alignment between a pair of polygons is assumed when the road feature nearest to one of the polygons is the same road that is nearest to the other. The reason for a road alignment test is illustrated by the example configuration shown in Figure 6.9. There are of course deficiencies with such a simplistic approach. For example, it is possible to imagine situations in which a

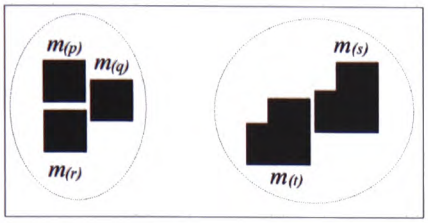
pair of object have a similar number of edges, but which are dissimilar in shape (and vice versa). In addition, a pair of polygons sharing the same nearest road will not necessarily be aligned. However, experiments presented later suggest that, when applied to OS™ MasterMap® data used for this research, the function works well in most situations. Furthermore, it is noted that the main emphasis of the work presented in this section is to test and evaluate the application of SA to high-order features, as opposed to the development of a sophisticated clustering algorithm.

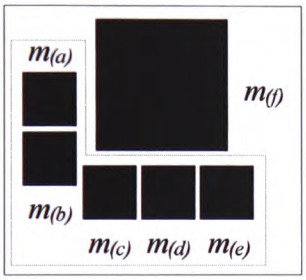
```
function GroupPolygons
  add all polygons to polygon-list
  while polygon-list not empty
      create new-polygon-group
      take a-polygon from polygon-list
      while a-polygon !=NULL
            add a-polygon to a new-polygon-group
            a-polygon = FindNextPolygon(polygon-list, new-polygon-group)
      endwhile
      store new-polygon-group
   endwhile
```

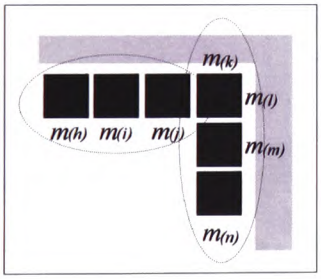
Figure 6.7: Function GroupPolygons.

Function GroupPolygons identifies high-order features by *growing* each group outward from an initial arbitrarily chosen starting polygon. The function *FindNextPolygon* searches the list of polygons that do not yet belong to a group for a polygon that meets the grouping criteria required in order for it to be added to the current group. The first such polygon encountered is selected, removed from the list and returned. If no polygon meets the criteria, a NULL value is returned.









Distance:

Distance refers to the minimum separation between the edges of a pair of objects. Pairs of objects which lie within the specified minimum separating distance of each other are candidates for grouping.

Shape:

The shape of an object and hence its complexity is measured by the number of edges it contains.

Size:

Size is calculated as the area of the object. Smaller buildings are grouped separately to larger buildings.

Alignment to road:

Buildings which share the same nearest road feature and lie within a specified minimum distance of that feature are candidates for grouping. Without this test, the situation illustrated in Figure 6.9 could occur.

Figure 6.8: Criteria for grouping.

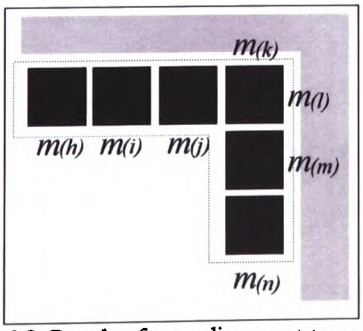
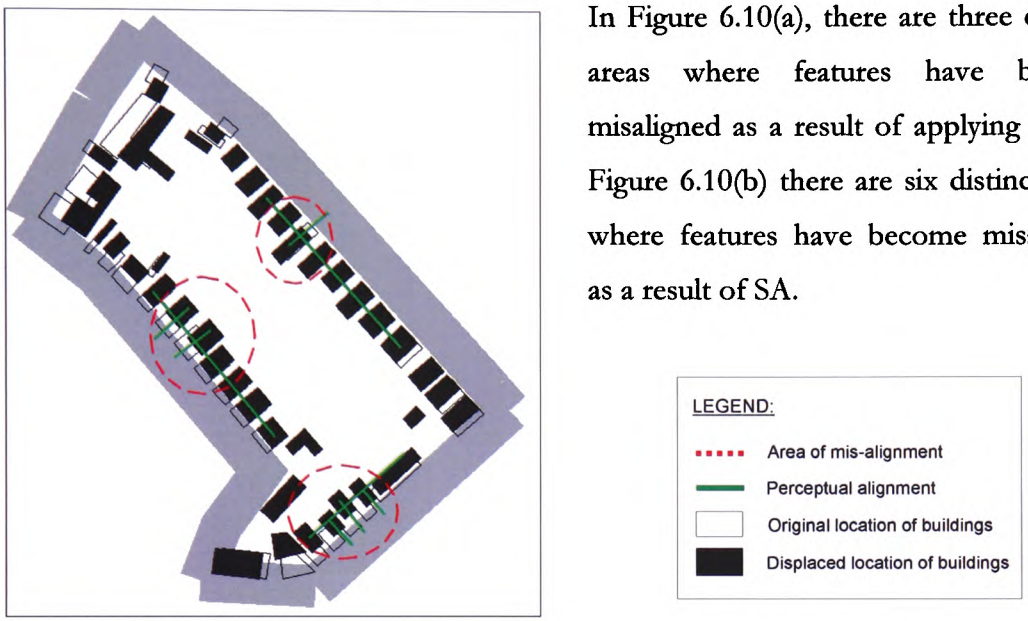


Figure 6.9: Result of non alignment to road test.

6.1.4 Evaluation

The first set of experiments involves making use of high-order features in a number of data sets where feature misalignment has occurred previously. Two displays highlighting such deterioration in feature alignment are shown in Figure 6.10.



In Figure 6.10(a), there are three distinct areas where features have become misaligned as a result of applying SA. In Figure 6.10(b) there are six distinct areas where features have become misaligned as a result of SA.

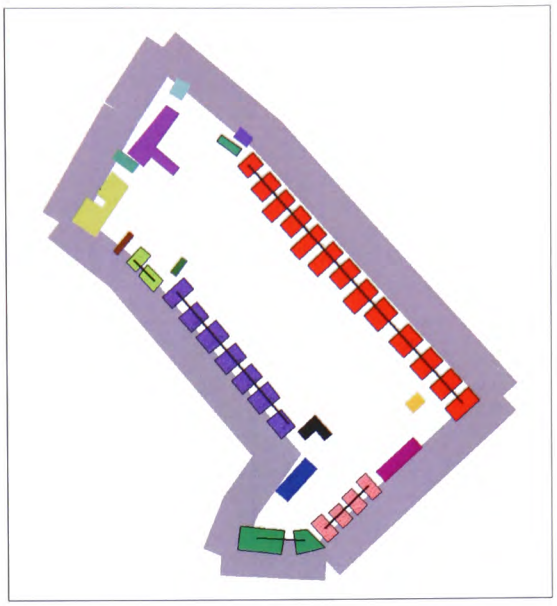
(a)

(b)

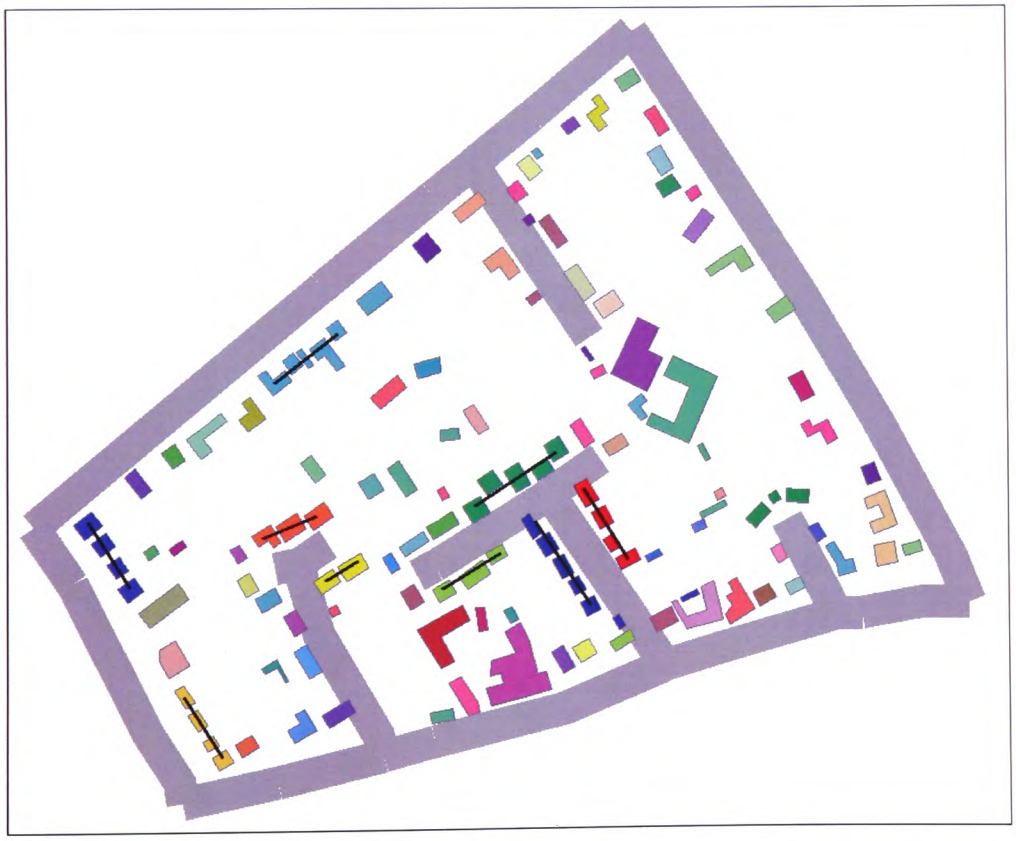
Figure 6.10: Examples of feature misalignment, post generalisation.

6.1.4.1 Manually Generated Groups

The first experiment relies on manually generated groups (see Figure 6.11). For easier representation, the following diagrams have been colour coded so that each group can be identified. Straight lines aid in visualising connections between map objects for each group.



(a)

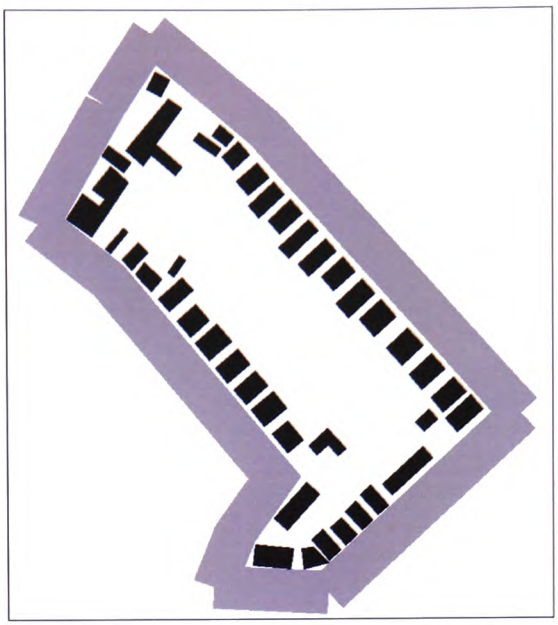


(b)

Figure 6.11: Manual group assignment.

6.1.4.2 Applying SA to Manually Generated Groups

Figure 6.12 shows the output display when SA is applied to manually generated high-order features. It can be seen that objects that previously became misaligned (see Figure 6.10) now retain their alignment.



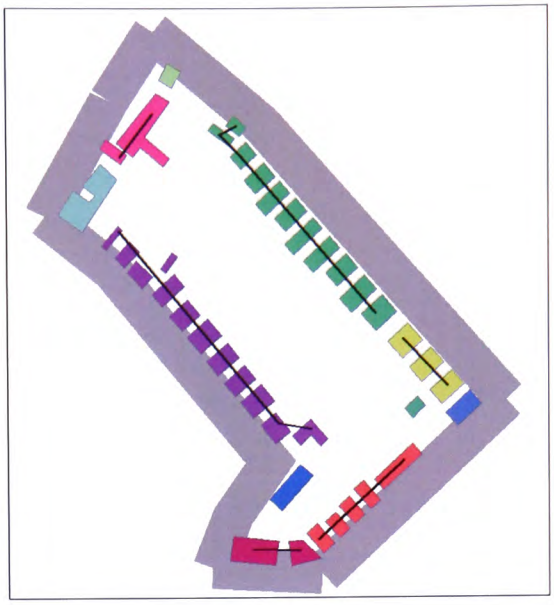
(a)

(b)

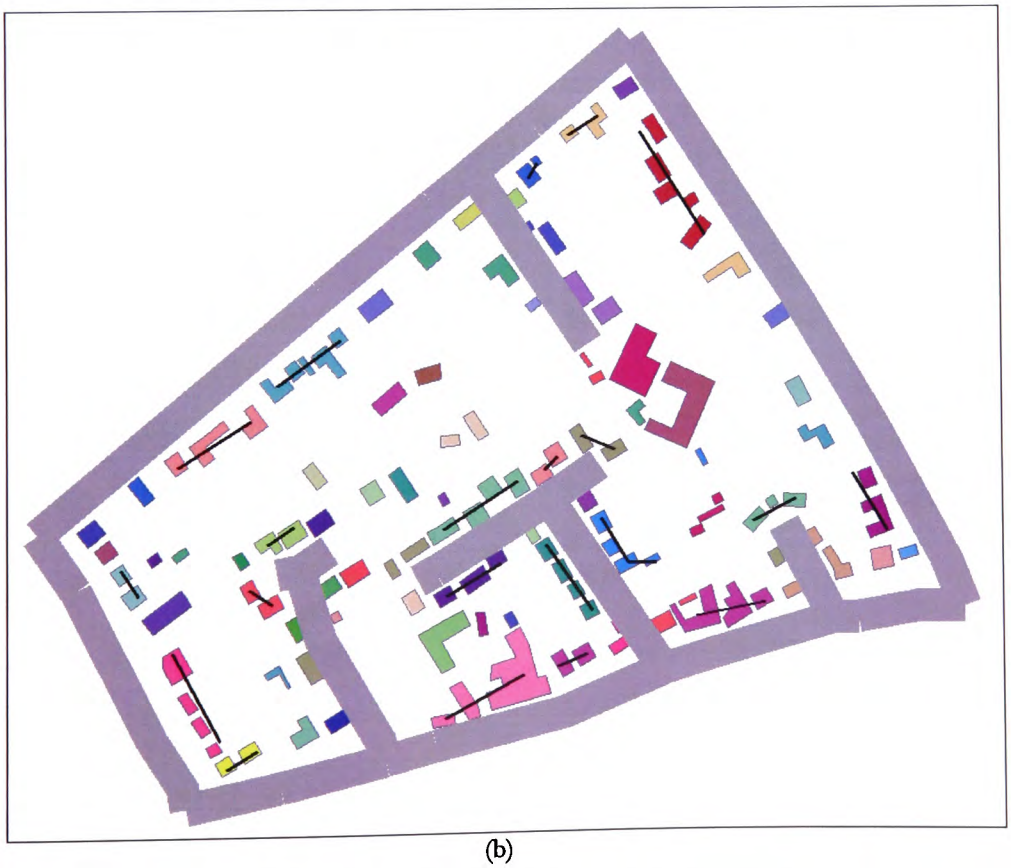
Figure 6.12: Results of running SA on manually generated groups.

6.1.4.3 Automated Grouping

The second stage of the experiment makes use of the grouping function to group objects together based on distance, size, shape and alignment to road (Figure 6.13). In both cases the distance threshold to which objects are grouped is set to 6 metres. The experiment makes use of a C++ implementation of the grouping function. The automatic grouping function is assessed by visually comparing its results against manual groupings. However, in future development it will be beneficial to include a quantitative assessment (by comparing measures such as total number of groups, total number of misaligned objects etc).



(a)

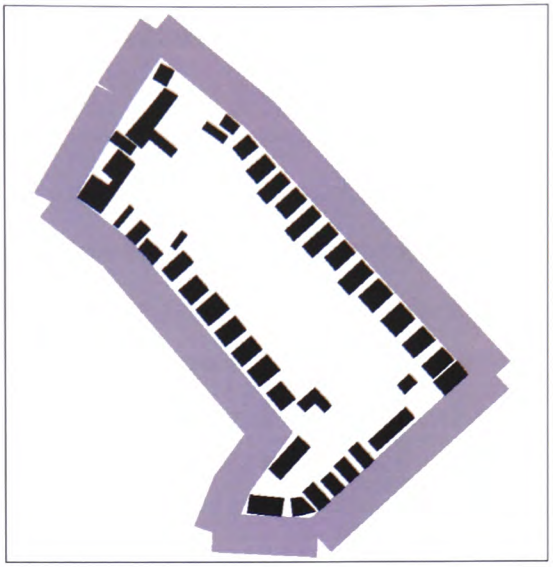


(b)

Figure 6.13: Group assignment using grouping function.

6.1.4.4 Applying SA to Automatically Generated Groups

Figure 6.14 shows the output produced by applying SA to high-order features generated from the grouping function. When compared against Figure 6.12, the use of the grouping function alongside SA is shown to preserve feature alignment. When compared against one another, outputs from both manual and automated techniques produce similar results (see Figure 6.12 and 6.14).



(a)

(b)

Figure 6.14: Results of running SA on automatically generated groups.

6.1.5 Execution Time Improvements

Chapter 3 discussed the principle operation of SA, where graphic conflict is resolved using a trial position and cost function strategy. For small-sized data sets, the time taken to arrive at an optimal solution can be relatively short. However, for larger-sized data sets the time taken can be considerably longer and in most cases, unsuitable for use on real-time applications. A partial solution to this problem is inherent within high-order features and is related to the decreased number of individual objects handled by SA due to object clustering. This results in a significant reduction in the search space which is generated and evaluated. Results reported below (see Table 6.1) suggest an overall decrease in execution times of ~50% when SA is applied to high-order data sets. For the sake of a valid comparison, the results are compared to the ones obtained in Chapter 4.

	Best-so-far results		*Results using object grouping*
Data set	*Conflict value*	*Execution time (secs)*	*Execution time (secs)*
1	146400	79.194	43.584
2	40600	7.525	2.964

Table 6.1: Increase in performance using high-order features.

6.2 Continuous Search Space

The SA work presented up to this point has made use of fixed trial positions for the modification of areal objects for the purpose of graphic conflict resolution. This approach has shown to be of use. However in some circumstances, the use of a discrete search space can lead to problems. For example, consider Figure 6.15, in which the building feature in (a) is in conflict with a symbolised road feature.

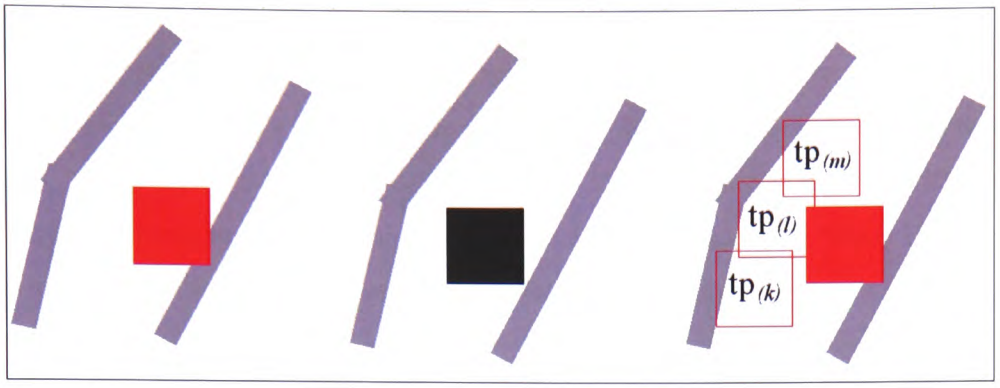


Figure 6.15: Conflict between areal and linear feature (a), desired result (b), limitations of discrete search space (c).

It can be seen in (b) that displacing the building slightly to the left would resolve all graphic conflict. Unfortunately, a trial position *'k'* corresponding to such a displacement does not exist; in fact, displacing the building to any of its available trial positions results in further conflict (c). A similar, if perhaps less critical, problem is illustrated in Figure 6.16. As before, graphic conflict (a) can be resolved by displacing the building feature slightly to the left, but, again, a trial position corresponding to such a displacement does not exist. This time graphic conflict can be resolved by making use of one of the available trial positions (b); however, the problem now is that the displacement appears excessive. The same problem can occur when applying the reduction operator, where map objects are reduced in size in a way that appears excessive.

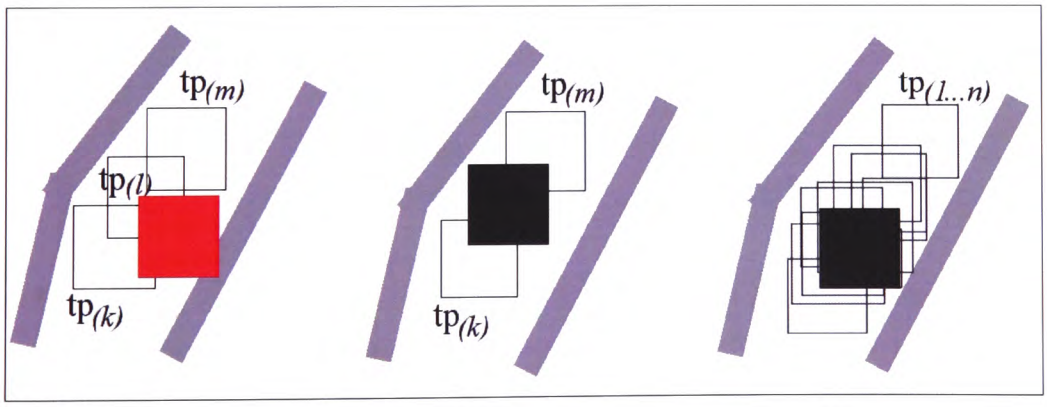


Figure 6.16: There exists a trial position to which the object can move (a). However this causes excessive displacement to occur (b). A possible solution would be to generate more trial positions into which the object can move (c).

One solution to the problems described above is to increase the resolution of the search space by adding to the number of displaced and reduced trial positions (see Figure 6.16c).

However this approach has two potential limitations. First, it is clear that however many trial positions which are added, the search space remains discrete and there is no guarantee that in any given situation the appropriate displacement or reduction will exist as a trial position. Second, increasing the number of trial positions results in an increase in size of the search space and this could lead to increased execution times. Therefore an alternative solution that has been adopted involves the use of a continuous search space.

A continuous search space approach has been suggested previously by Strijk and van Kreveld (2002) for the purpose of point labelling. In the context of the SA solution, the concept of using a continuous search space is quite straightforward and involves the generation of random trial positions on the fly, as opposed to the use of pre-generated fixed trial positions. The SA algorithm will work in very much the same way as before, with its main loop again beginning by choosing a modifiable object at random. However, instead of then choosing a trial position *'k'* at random, the next step will involve the selection of an operator at random (i.e. displace, reduce, enlarge or delete) together with appropriate randomly generated operator parameters (i.e. if displace is selected then a random displacement vector is generated, and if reduce is selected then a random reduction factor is generated). The operator, and parameters, is then applied and the algorithm proceeds as before (see Figure 6.17). By adopting the use of a continuous search space, results demonstrate that the limitations associated with a discrete solution space can be overcome to produce improved graphic display output (see Figure 6.18 and Figure 6.19).

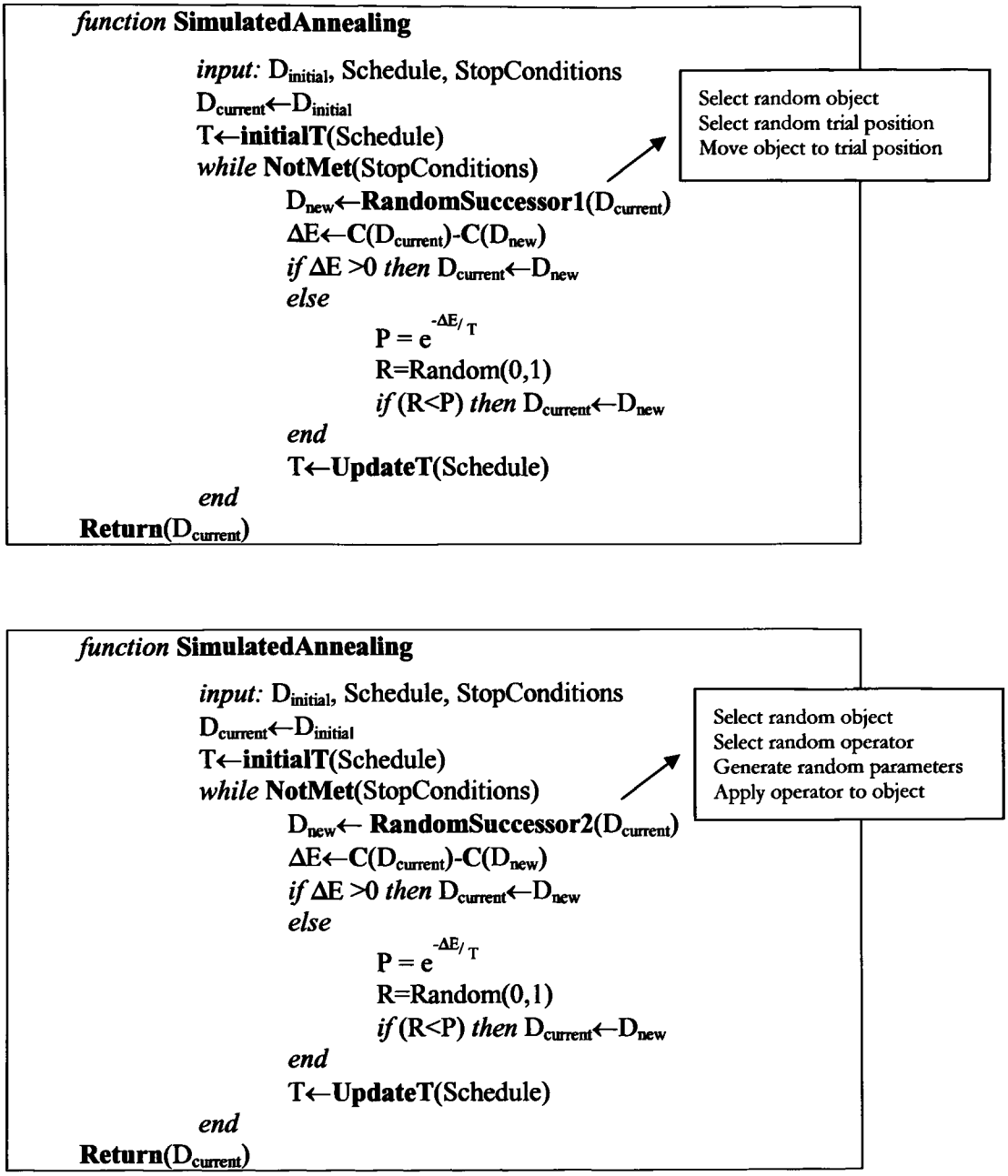


Figure 6.17: Pseudo code implementation for discrete search space (top) and continuous search space (bottom).

6.2.1 Continuous Search Space Results 1

Figure 6.18: Example 1. Comparison between discrete (left) and continuous search space (right).

6.2.2 Continuous Search Space Results 2

Figure 6.19: Example 2. Comparison between discrete (left) and continuous search space (right).

6.3 Additional Feature Classes

One of the requirements set out by the OS™ was that the SA implementation be able to handle additional feature classes, with each having their own independent constraints. Previously, only roads and buildings were supported, with each having their own set of generalisation parameters. So a further refinement to the original SA is the incorporation of additional feature classes (in the form of railways, rivers, woodland, farmland and lake/pond areas). Implementation to include the extra classes was not difficult and turned out to be little more than a programming exercise. More interesting however, was the question of whether or not the SA software would cope with the added problem complexity.

The results exceeded initial expectations and that SA does indeed handle the additional feature classes, each with their own minimum distance clearances (see Table 6.2). Output displays (both cartographic and computational performance) are in-line with those produced when limited to just two classes. Figure 6.20 and Figure 6.21 presents the output produced after applying SA to various feature classes using values supplied from Table 6.2. SA took approximately 4.2 seconds to execute for example 1 and 2.9 seconds to execute for example 2.

Example 1 Feature class type	Min. distance clearances (in metres)
Road network	10
Railway line	15
River	20
Buildings	1
Lake/Ponds	9
Woodland	9

Example 2 Feature class type	Min. distance clearances (in metres)
Road network	10
River	10
Buildings	1
Woodland	3
Farmland	2

Table 6.2: Minimum distance clearances for various feature classes.

6.3.1 Additional Feature Classes Results 1

Figure 6.20: Example 1. Additional feature classes in SA (railway lines, roads, woodland areas, lakes and buildings).
Conflict between various feature classes (top), and result of running SA using values supplied from Table 6.2 (bottom).

6.3.2 Additional Feature Classes Results 2

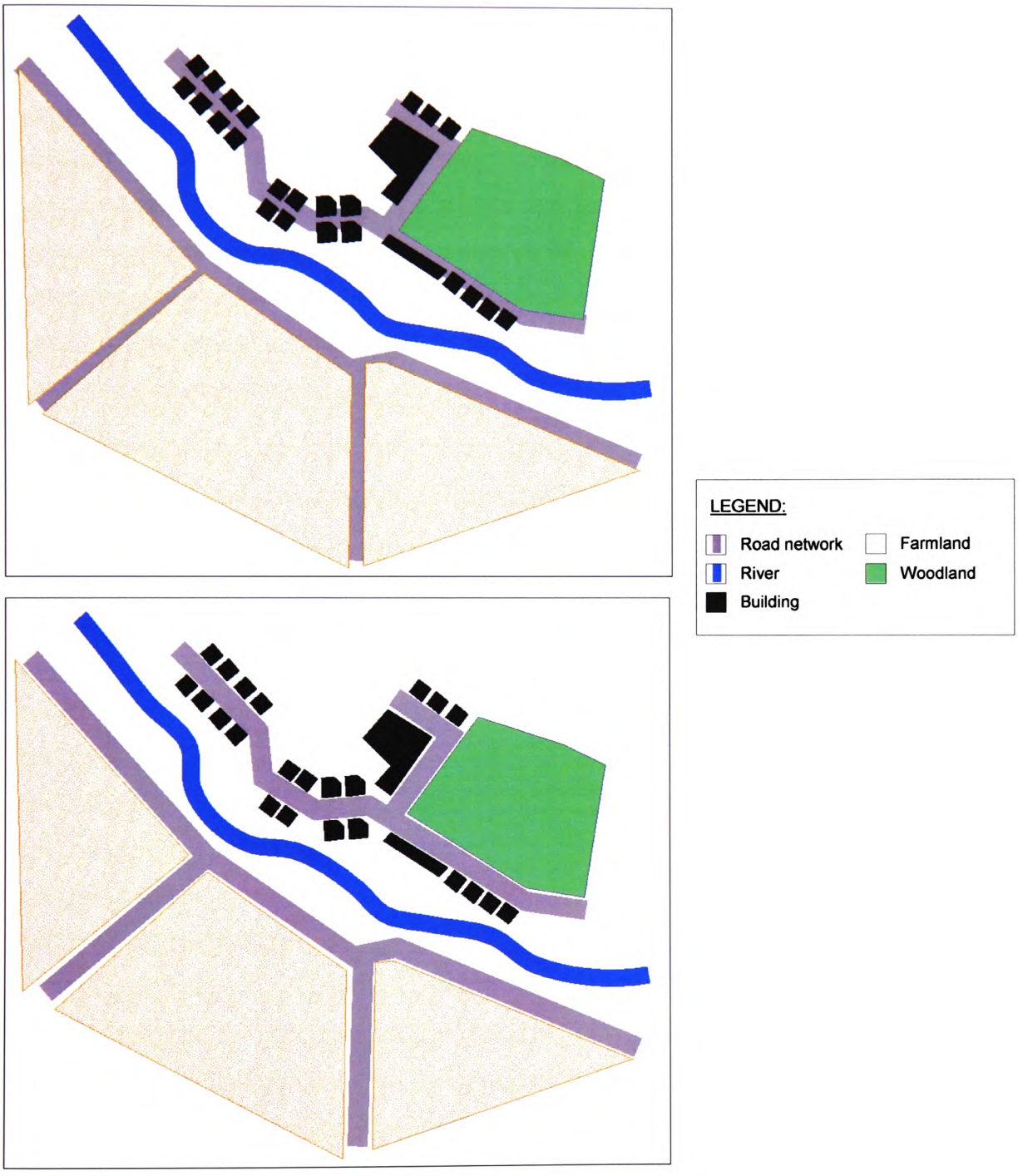


Figure 6.21: Example 2. Additional feature classes in SA (roads, woodland areas, farmland and buildings).
Conflict between various feature classes (top), and result of running SA using values supplied from Table 6.2 (bottom).

6.4 Summary

This chapter details research related to refining the SA approach for the purpose of producing improved graphic-display output. One key conclusion that had earlier been reached was the need to maintain feature alignment during the generalisation process, as it is possible in some situations for groups of map objects, representing high-order features to become misaligned, thus resulting in a deterioration of map quality. In order to prevent such deterioration from occurring, the research in this chapter focuses upon object clustering techniques using both manual and automated strategies. This chapter has also investigated the limitations associated with the use of a discrete search space and it offers a practical solution in the form of a continuous search space version of the SA algorithm. The chapter concludes with a discussion on how additional feature classes have been incorporated into the SA solution.

Chapter 7
Discussion

This chapter discusses the research presented in the thesis, bringing together the conclusions reached. Limitations of the work are also discussed and suggestions for future work articulated.

7.1 Aims and Achievements

The general aim of research, as outlined in Chapter 1, has been to design, implement and evaluate automated procedures to assist in the process of cartographic map generalisation. The specific question asked at the outset of the work was: "To what extent can an optimisation technique, such as SA, be used as a process control to automate the generalisation process?" In order to answer this question, the research has addressed three main areas (as outlined in 1.9):

1. To carry out a thorough evaluation of the existing SA technique.
2. To assess the suitability of SA as a means of a process control to assist in cartographic generalisation.
3. Design, implement and evaluate new techniques to overcome the limitations of SA and integrate these into a new generalisation framework.

The research has achieved five key advancements over the original SA technique. These are:

1. Reduced execution times.
2. A greater support for generalisation operators.
3. Resolved issues related to maintaining high order feature alignment.
4. A greater support for additional feature classes.
5. Refinements to the search space.

Chapters 4 through 6 provide evidence of work relating to these areas, building upon previous work on SA by Ware and Jones (1998). An account of their work was presented in Chapter 3.

7.1.1 Reduced Execution Times

A series of techniques have been developed to reduce execution times, since this had been identified as a limitation of the original SA work. This is of particular importance as execution times for the original implementation were too slow for it to be of practical use on large data sets and in applications requiring on-the-fly generalisation (e.g. location based services and in-car navigation systems etc). Results published demonstrate that the techniques devised offer reduced execution times.

The use of data partitioning in 4.3 for dividing the map data into regions allows the annealing process to meet more appropriately the specific demands of each region. The process of partitioning the map data acts as a non-stationary model (Brunsdon 1996, Perrin 1999) - as the annealing values used for each segment varies spatially. This leads to a reduction in execution times, and a lower final conflict value. This methodology was evaluated in a series of experiments using a combination of manually and automatically generated annealing values. The results in 4.4.2 clearly demonstrate that automated generation of the initial annealing values results in a shorter execution time as well as a lower index of conflict in the resultant data set.

Further improvements on the execution time can be made by combining data partitioning with a two-stage annealing process which is based on an idea presented by Varanelli and Cohoon (1993). Results presented in this thesis suggest two-stage annealing to be superior to single stage annealing, both in terms of execution times and the amount of conflict resolved.

The research has demonstrated that various methods can be employed to reduce execution times of the SA implementation whilst achieving greater conflict resolution.

7.1.2 A Greater Support for Generalisation Operators

Chapter 5 discussed the fact that the use of the displacement operator in isolation cannot guarantee that all conflict will be resolved. A solution to this limitation has been presented in which additional operators are integrated into the original implementation. The additional operators are object deletion, object size reduction and object size enlargement. The technique used for including the extra operators involves assigning each modifiable feature additional trial positions that correspond to its deleted, reduced and enlarged states.

In order to provide a flexible and dynamic approach in applying these additional operators, appropriate cost values are assigned to each operator which acts as a form of process control. These costs are used as a means of prioritising the use of the various operators. In addition a 'building importance factor' in 5.4 plays an important role when dealing with multiple operators. Generalisation in SA is no longer regarded as simply displacing or reducing objects arbitrarily in an attempt to resolve conflict but instead greater consideration is now given to the relative importance of objects which adds a level of intelligence – an important factor missing from the original implementation. A practical experiment highlighted the use of the 'building importance factor' when dealing with the deletion operator - whereby smaller non-important buildings are removed during the generalisation process to free up extra map space for more important buildings. Maps generated from the second data set demonstrate the usefulness of this approach, in which all conflict has been resolved in a manner that would be considered consistent with a manually generalised approach.

Overall, the use of SA in conjunction with multiple generalisation operators, the use of weighting and a 'building importance factor', has further reduced conflict during the generalisation process.

7.1.3 High-order Feature Alignment

A further shortcoming of the original SA implementation is in the deterioration of map quality in situations where related map features become misaligned as a result of generalisation. To overcome this problem a technique has been devised that groups related objects together according to a series of criteria. The SA operators are

subsequently applied to object groups. This method offers several advantages, including improving execution times and, more importantly, preserving spatial relationships in the form of maintaining feature alignment between map objects.

This research has seen the development of a grouping function that acts independently and as a pre-process to SA. This allows the user to check the assignment of groups prior to performing generalisation using SA. A disadvantage of the current approach is that conflict may be present between buildings prior to object grouping. This results in conflict being retained within the group itself. A solution to this problem might be to run the SA algorithm locally within each group as a post-process.

An evaluation of the grouping function developed was made by comparison with manual grouping that makes use of conventional perceptual methods. Both approaches produced similar results, although in some cases the automated grouping function is not entirely of use at optimally clustering objects together. This is simply a consequence of the limitations placed upon the further development of the automated grouping function by the time and scope constraints of this research.

One unexpected advantage to be gained from grouping map features together is related to what is best described as a remote solution. For example, consider a map display *D* (see Figure 7.1a) made up from a series of polygonal objects *M* and two fixed linear objects, *F*. In the original implementation, object *M(i)* would always remain unresolved (see Figure 7.1b), as the only solution to resolving this conflict would be for all surrounding objects *M(j...m)* to be displaced individually one at a time away from *M(i)*, freeing up sufficient map space to allow object *M(i)* to move. In the current implementation, due to the randomness of the algorithms execution, object *M(i)* would in all cases remain unresolved. When dealing with high order features, objects *M(i...m)* are grouped and treated as a single entity. Displacement is then performed ***en masse*** and all objects are displaced including object *M(i)* (see Figure 7.1c).

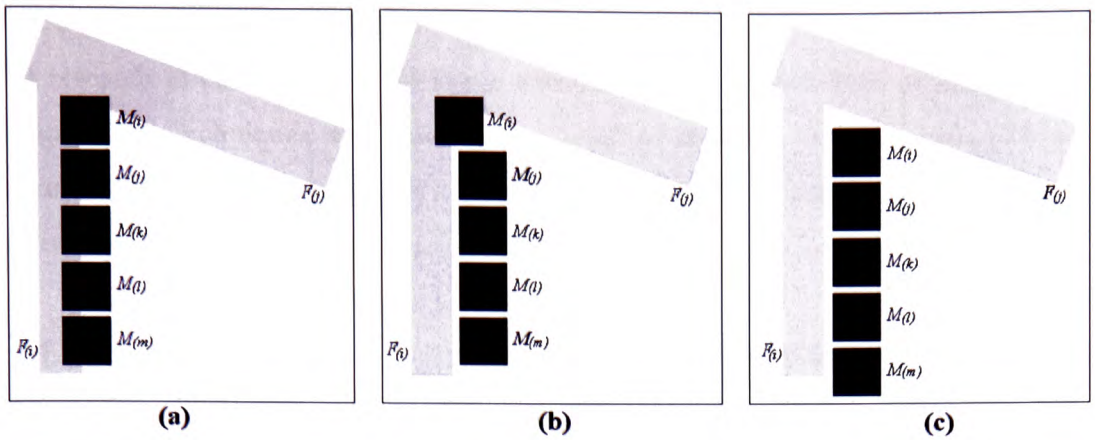


Figure 7.1: Remote solutions in SA.

7.1.4 Continuous Search Space

The use of a continuous search space in 6.2 produces maps with less gross distortion when compared to a discrete search space solution. The use of a continuous search space prevents objects being displaced from their original location in a way that appears excessive. Additionally, for objects that require a reduction or enlargement in size, this process is undertaken gradually – whereby the reduction or enlargement of an object varies in size only to the degree to which it is required. The benefits of utilising a continuous search space was only realised during the later part of the research, as greater precedence was directed towards that of reducing execution times and incorporating additional operators in the original SA framework.

7.1.5 A Greater Support for Additional Feature Classes

A final area of refinement that has been made to the original implementation is in the incorporation of additional feature classes. This was achieved using simple modifications to the existing implementation. The experiment in 6.3 made use of additional feature classes (i.e. railway lines, rivers etc) each having their own independent generalisation constraints in the form of minimum distance clearances. The purpose of this experiment was to determine how capable the SA algorithm was in handling the added complexity of additional feature classes.

7.2 Future Work

The research documented in this thesis demonstrates the capabilities of using iterative advancement techniques for ameliorating conflict in scale-reduced maps. There is enormous scope for further work in this field. The following is a list of suggestions for future work that flow from the findings of this research:

1. There is clear evidence that further work is needed in exploring alternative strategies for improving execution times as it could still be argued that, despite the results presented within this research, execution times remain too slow for use with on-the-fly generalisation tasks. One area not investigated as part of this research was work undertaken on Parallel Simulated Annealing (Varanelli, 1996). Other techniques for improving execution times include exploring alternative iterative improvement algorithms (e.g. Genetic Algorithms and Tabu search). Some initial work has already been carried out by researchers in the School of Computing, University of Glamorgan (Wilson *et. al* 2003, Ware *et. al* 2003, Ware *et. al* 2002). It would be interesting to see how SA compares against alternative iterative improvement algorithms, both in terms of execution times and the amount of conflict resolved.

2. It would be interesting to investigate the incorporation of more sophisticated operators into SA. The use of additional operators has proven highly effective in reducing conflict. Additional operators to enhance SA further may include typification and simplification of map features.

3. Another area for further study is to determine whether SA can be adapted for use with the generalisation of linear features (e.g. rivers, roads etc). Currently only generalisation related to areal objects (i.e. such as buildings) is handled. For example, in circumstances where two linear features are in conflict as a result of scale-reduction and symbolisation, points along the lines that share in conflict might be displaced using a trial position strategy (see Figure 7.2, below).

4. It would be interesting to see the outcome of applying the current implementation within a variety of larger or alternative generalisation systems. For example, incorporating SA as a method made available to an Agent based

system or integrating SA into a commercial system such as ESRI's™ ARCMAP™ software.

5. A final area of work for future study is the need to develop a more sophisticated building grouping function, possibly incorporating work on building clustering by Christophe and Ruas (2002). Their method offers a far more comprehensive solution for clustering objects than the implementation presented in this research. The complexity of integrating their work was beyond the time constraints of this research.

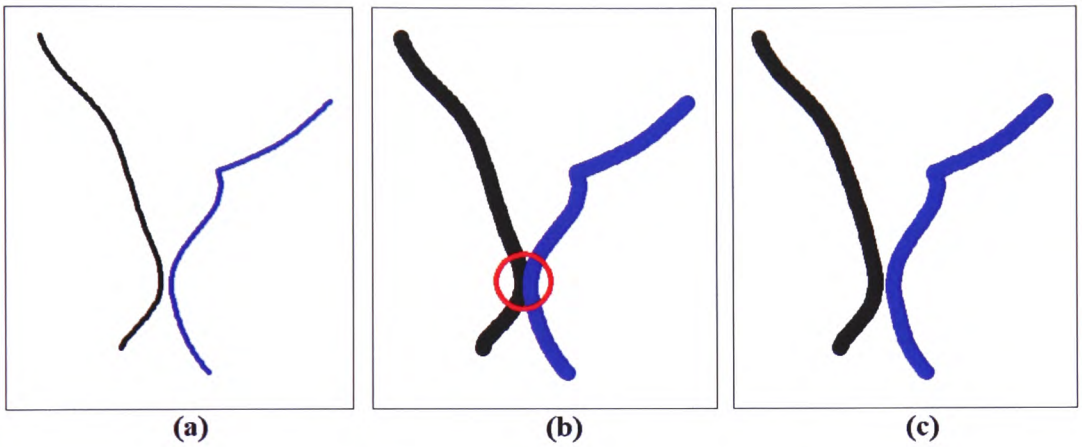


Figure 7.2: Line generalisation using SA. Original map (a); overlap of two linear features as a result of scale-reduction and symbolisation (b); points along the area of conflict are displaced using SA and the line redrawn(c).

Chapter 8
Conclusion

This research was an attempt to design, implement and evaluate automated procedures to assist in the process of cartographic map generalisation. More specifically, the research question asked "To what extent can an optimisation technique, such as SA, be used as a process control to automate the generalisation process?". In order to answer this question, the research has addressed three main areas (as outlined in 1.9):

1. To carry out a thorough evaluation of the existing SA technique.
2. To assess the suitability of SA as a means of a process control to assist in cartographic generalisation.
3. Design, implement and evaluate new techniques to overcome the limitations of SA and integrate these into a new generalisation framework.

In evaluating the existing SA technique, a number of problems were found and this research attempted to correct those.

Execution times for the original SA algorithm were impractical, and a number of techniques were developed to make the algorithm more efficient. Data partitioning and the use of a two-stage annealing process both reduced the execution times and decreased the amount of final conflict particularly when used together.

It was evaluated that the number of operators available to SA were insufficient to completely resolve all graphic conflict. As a result a number of new operators were incorporated. Object deletion, object size reduction and object size enlargement operators were developed and applied with a modified schedule which took into account a new factor, termed "building importance". The use of these new operators along with the building importance attribute allowed a greater amount of graphic conflict to be resolved.

An undesirable side-effect of the original SA process was in the disruption of high-order features. This resulted in features of the map becoming unrecognisable following the generalisation process. In order to combat this, a technique was developed that groups objects together based on a series of test criteria. SA is then applied to each group. This grouping function is run as a pre-process, and excellent results were achieved in the maintenance of these high-order features compared against their original alignment. It was noted however, that any conflict within the groups themselves was not resolved, and it was suggested that perhaps a generalisation process should be applied to the contents of each group individually following the generalisation of the map as a whole. It was noted that the simple automatic grouping function developed in this research did not group as well as a manual approach, however the optimisation of such a function was beyond the scope of this project. Further research in this area should be able to provide a more efficient algorithm.

An unexpected advantage of grouping of high-order features is that of a remote solution. This allows an object in conflict which is effectively hemmed in by a number of other objects, to move out of conflict with a single operation. Previously graphic conflict could only be attained by a complex series of operations logically laid out, something that the random applications within SA would be unlikely to complete.

In addition to the reduction of execution times, an attempt was made to increase the gross resolution of graphic conflict. It was discovered late on in the project that the use of a continuous search space provided excellent graphic conflict resolution surprisingly without unduly increasing the execution time of the algorithm. This also resulted in far less gross distortion of the map or objects post-generalisation.

Finally, a successful attempt was made to modify the SA process to cater for additional feature classes such as linear features (roads, railway lines) and irregular areal objects (e.g. ponds, lakes). Experiments showed that SA could successfully be used to accommodate additional feature classes.

Final Remarks

The work presented in this thesis has demonstrated the potential of iterative improvement algorithms as a means of reducing graphic conflict in scale-reduced maps. Feedback from the OS™ on the practical usage of this research has been extremely positive. Although the research has tackled key issues relating to map generalisation, SA is by no means considered a complete solution for automating the generalisation process, but instead is viewed more as a generalisation tool possibly for use within a larger generalisation framework (e.g. such as the AGENT project).

References

Ai, T. and van Oosterom, P. (2002) "Displacement methods based on field analysis", *Joint ISPRS-ICA Workshop on Multi-Scale Representations of Spatial Data*, Ottawa. Available from: http://www.geo.unizh.ch/ICA/docs/workshops/ottawa2002.html

Bader, M., Barrault, M., Regnauld, N., Mustière, S., Duchêne, C., Ruas, A., Fritsch, E., Lecordix, F., and Barillot, X. (1999) "AGENT work package D2 – Selection of basic algorithms technical report", Department of Geography, University of Zurich.

Bader, M. (2001) *Energy minimization methods for feature displacement in map generalization,* PhD Thesis, University of Zurich, Switzerland.

Bader, M. and Weibel, R. (2004) "Building displacement over a ductile truss", to appear in the *International Journal of Geographic Information Systems: Special Issue on Map Generalisation.*

Beard, K. (1991) "Generalization operations and supporting structures", in *ACSM-ASPRS Annual Convention, Auto-Carto 10,* Baltimore, USA, **6**, 29-45.

Bobrich, J. (1996) *Ein neuer Ansatz zur Kartographischen Verdrängung auf der Grundlage eines mechanischen Federmodells,* PhD Thesis, Deutsche Geodätische Kommission, Műnchen, Reihe C.

Brassel, K. and Weibel, R. (1988) "A review and conceptual framework of automated map generalization", *International Journal of Geographic Information Systems*, **2**(3), 229-244.

Brunsdon, C., Fotheringham, A. S. and Charlton, M. (1996) "Geographically weighted regression: A method for exploring spatial non-stationarity", *Geographical Analysis*, **28**, 281-298.

Bundy, G. L. (1996) *Automated cartographic generalization with a triangulated spatial model,* PhD Thesis (available from The British Library).

Burghardt, D. and Meier, S. (1997) "Cartographic displacement using the snakes concept", *Semantic Modelling for the Acquisition of Topographic Information from Images and Maps,* Birkhaeuser-Verlag, Basel, 59-71.

Buttenfield, B.P. and McMaster, R. B., Eds. (1991) *Map Generalization: Making Rules for Knowledge Representation*, Longman Group, London.

Chew, L.P. (1989) "Constrained Delauney triangulations", *Algorithmica*, 4, 97-108.

Christensen, J., Marks, J. and Shieber, S. (1995) "An empirical study of algorithms for point-feature label placement", *ACM Transactions on Graphics*, **14**(3), 203-232.

Christensen, A.H.J. (1999) "Cartographic line generalisation with waterlines and medial-axes", *Cartography and GIS*, **26**(1), 19-32.

Christophe, S. and Ruas, A. (2002) "Detecting building alignments for generalisation purposes", in *Proceedings of 10th International Symposium on Spatial Data Handling*, Ottawa, 419-432.

Chrobak, T. (2000) "A numerical method for generalising the linear elements of large-scale maps, based on the example of rivers", *Cartographica*, **37**(1), 49-55.

de Berg, M., van Kreveld, M. and Schirra, S. (1995) "A new approach to subdivision simplification", in *Proceedings of AUTO-CARTO 12*, **4**, 79-88.

de Berg, M., van Kreveld, M. and Schirra, S. (1998) "Topologically correct subdivision simplification using the bandwidth criterion", *Cartography and GIS*, **25**(4), 243-257.

DeFloriano, l. and Puppo, E. (1988) "Constrained Delauney triangulation for multi-resolution surface description", in *Proceedings of the 9th International Conference on Pattern Recognition*, Rome, Italy, 566-569.

DeLucia, A. and Black, R.T. (1987) "A comprehensive approach to automatic feature generalisation", in *Proceedings of 13th International Cartographic Conference*, Mexico, 173-191.

Douglas, D.H. and Peucker, T.K. (1973) "Algorithms for the reduction of the number of points required to represent a line or its caricature", *The Canadian Cartographer*, **10**(2), 112-122.

Dowsland, K.A. (1995) "Simulated annealing" in *Modern Heuristic Techniques for Combinatorial Problems*, McGraw-Hill, 20-69.

Freeman, H. (1991) "Knowledge classification and organization", in *Map Generalization: Making Rules for Knowledge Representation*, B.P. a. M. Buttenfield R.B., Eds. Longman Group, London, 86-102.

Galanda, M. (2001) "Optimization techniques for polygon generalization", in *Proceedings of the 4th ICA Workshop on Progress in Automated Map Generalization*, Beijing, Available from: http://www.geo.unizh.ch/ICA/docs/workshops/beijing2001.html

Gelfand, S.K. and Mitter, S.K. (1985), "Analysis of simulated annealing for optimization", in *Proceedings of the 24th Conference on Decision and Control*, Ft. Lauderdale, FL, 779-886.

Haire, K.R. and Hardy, P.G. (2001) "Active agent based approaches to automated generalisation", in *Proceedings of 9th Annual GISRUK Conference*, University of Glamorgan, 319-320.

Hajek, B. and Sasaki. (1988) "Simulated annealing-To cool or not", *System and Control Letters*, **12**, 443-447.

Hajek, B. (1988) "Cooling schedules for optimal annealing", *Mathematics of Operations Research*, **13**(2), 311-329.

Harrie, L.E. (2000) "The constraint method for solving spatial conflicts in cartographic generalisation", *Cartography and GIS*, **26**(1), 55-69.

Harrie, L. and Sarjakoski, T. (2002) "Simultaneous graphic generalisation of vector data sets", *Geoinformatica*, **6**(3), 233-261.

Højolt, P. (2000) "Solving space conflicts in map generalisation: Using a finite element method", *Cartography and GIS*, **27**(1), 65-73.

Ingber, L. (1993) "Simulated annealing: Practice versus theory", *Mathematics and Computer Modelling*, **18**(11), 29-57.

Jones, C.B. and Abraham, I.M. (1987) "Line generalisation in a cartographic database", *Cartographica*, **24**(3), 32-45.

Jones, C.B. (1990) "Conflict resolution in cartographic name placement", *Computer and Design*, **22**(3), 173-183.

Jones, C.B., Bundy, G.L. and Ware, J.M. (1995) "Map generalisation with a triangulated data structure", *Cartography and GIS*, **22**(4), 317-331.

Kass, M., Witkin, A., Terzopoulas, D. (1987) "Snakes: Active contour models", in *Proceedings of the First International Conference on Computer Vision*, 259-268.

Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. (1983) "Optimization by simulated annealing". *Science*, **220**(4598), 671-680.

Kohonen, T. (1982) "Self-organized formation of topologically correct feature maps", *Biological Cybernetics*, 43, 59-69.

Lamy, S. *et al* (2000) "The application of agents in automated map generalisation", in *Proceedings of 19th ICA/ACI Conference*, Ottawa, 60-169.

Lecordix, F., Plazanet, C. and Lagrange, J-P. (1997) "A platform for research in generalisation", *Geoinformatica*, **1**(2), 161-182.

Lonergan, M. E. and Jones, C. B. (1999) "The deformation and displacement of areal objects during automated map generalisation: an approach to conflict resolution through maximising nearest neighbour distances", *ICA commission on Map Generalisation working group*, University of Glamorgan. Available from: http://www.comp.glam.ac.uk/pages/staff/mlonerga/workshoppaper.html

Li, M., Zhou, S. and Jones, C.B. (2002) "Multi-agent systems for web-based map information retrieval", *Springer Lecture Notes in Computer Science 2478 (GIScience 2002)*, 161-180.

Li, Z. and Openshaw, S. (1992) "Algorithms for automated line generalization based on a natural principle of objective generalization", *International Journal of Geographical Information Systems*, **6**(5), 373-389.

Li, Z. and Su, B. (1996) "Algebraic models for feature displacement in the generalisation of digital map data using morphological techniques", *Cartographica*, **32**(3), 39-56.

Lichtner, W. (1979) "Computer-assisted processes of cartographic generalisation in topographic maps", *Geo-Processing*, **1**(1), 183-199.

Lonergan, M. and Jones, C.B. (2001) "An iterative displacement method for conflict resolution in map generalisation", *Algorithmica*, **30**(2), 287-301.

Mackaness, W.A. (1994) "An algorithm for conflict identification and feature displacement in automated map generalisation", *Cartography and GIS*, **21**(4), 219-232.

Mackaness, W.A. (1995) "Analysis of urban road networks to support cartographic generalisation", *Cartography and GIS*, **22**(4), 308-316.

Mackaness, W.A. and Mackechnie, G.A. (1999) "Automating the detection and simplification of junctions in road networks", *Geoinformatica*, **3**(2), 185-200.

Marks, J. and Shieber, S. (1993) "The computational complexity of cartographic label placement", *Technical Report, Center for Research in Computing Technology*, Harvard University.

McMaster, R. B. and Shea, K.S. (1992) "Generalization in digital cartography", *Association of American geographers*, Washington D.C.

Metropolis, N., Rosenbluth, A. Rosenbluth, M., Teller, A. and Teller, E. (1953) "Equations of state calculations by fast computing machines", *Journal of Chemical Physics,* **21**, 1087-1091.

Monmonier, M. (1982) "Computer-assisted cartography-Principles and prospects", Prentice Hall, New Jersey.

Monmonier, M. (1986) "Toward a practicable model of cartographic generalisation", in *Proceedings of AutoCarto*, London, **2**, 257-266.

Monmonier, M. (1987) "Displacement in vector and raster-mode graphics", *Cartographica*, **24**(4), 25-36.

Monmonier, M. (1989) "Interpolated generalization: cartographic theory for expert-guided feature displacement", *Cartographica*, **26**(1), 43-64.

Monmonier, M. (1991) "The role of interpolation in feature displacement", in *Map Generalization: Making Rules for Knowledge Representation*, B.P. Buttenfield, R. B. McMaster, Eds. Longman Group, London, 189-203.

Muller, J.C., Lagrange, J.P. and Weibel, R. (1995) "GIS and generalization-Methodology and practice", I. Masser, F. Salge, Eds., Taylor and Francis, London, **1**.

Nelder, J. A. and Mead, R. (1965) "A simplex method for function minimization", *Computing Journal*, 7, 308-313.

Nickerson, B.G. (1988) "Automated cartographic generalisation for linear features", *Cartographica*, **25**(3), 15-66.

Ormsby, D. and Mackaness, W. (1999) "The development of phenomenological generalisation within an object-oriented paradigm", *Cartography and GIS*, **26**(1), 70-80.

Perrin, O. and Iovleff, S. (1999) "Estimating a non-stationary spatial structure using simulated annealing", *Geocomputation '99*, Fredericksburg, VA, USA. Available from: http://www.geovista.psu.edu/sites/geocomp99/Gc99/028/gc_028.htm

Rayward-Smith, V.J., Osman, I.H., Reeves, C.R., Smith, G.D. (1996) "Modern heuristic search methods", John Wiley & Sons.

Regnauld, N. (1996) "Recognition of building clusters for generalisation", in *Proceedings of 7th International Symposium on Spatial Data Handling*, Delft, 1-13.

Regnauld, N. (1998) *Généralisation du bâti: Stucture spatiale de type graphe et representation cartographique*, PhD Thesis, University of Provence, Aix Marseille 1.

Regnauld, N. (2001) "Contextual building typification in automated map generalisation", *Algorithmica*, **30**(2), 312-333.

Robinson, A.H., J.L. Morrison, A.J. Muehrcke, S.C. Guptill and Kimerling, A.J. (1995) *Elements of cartography*, John Wiley.

Ruas, A. and Plazanet, C. (1996) "Strategies for automated generalisation", in *Proceedings of 7th International Symposium on Spatial Data Handling*, Delft, **1**(6.1-6.18).

Ruas, A. (1998) "A method for building displacement in automated map generalisation", *International Journal of Geographic Information Systems*, **12**(8), 789-803.

Russell, S. and Norvig, P. (1995) "Artificial intelligence: A modern approach", Prentice-Hall.

Rutenbar, R.A. (1989) "Simulated annealing algorithms: An overview", *IEEE Circuits and Devices Magazine*, **5**(1), 19-26.

Rytz, A., *et al.* (1977) "Cartographic Generalisation", *Cartographic Publication Series, Swiss Society of Cartography*, **2**.

Saalfeld, A. (1999) "Topologically consistent line simplification with the Douglas-Peucker algorithm", *Cartography and GIS*, **26**(1), 7-18.

Sester, M. (2003) "Optimising approaches for generalisation and data abstraction", to appear in the *International Journal of Geographic Information Systems: Special Issue on Map Generalisation.*

Shea, K.S. and McMaster, R. B.. (1989) "Cartographic generalization in a digital environment: When and how to generalize", in *Proceedings of 9th International Symposium on Computer-Assisted Cartography (Auto-Carto 9)*, ASPRS/ACSM, 56-67.

Shea, K.S. (1991) "Design considerations for an artificially intelligent system", in *Map Generalization: Making Rules for Knowledge Representation*, Longman, 3-20.

Strijk, T. and van Kreveld, M. (2002) "Practical extensions of point labelling in the Slider model.", *Geoinformatica*, **6**(2), 181-197.

Su, B., Li, Z., Lodwick, G. and Müller, J-C. (1997) "Algebraic models for the aggregation of area features based upon morphological operators", *International Journal of Geographic Information Systems*, **11**(3), 233-246.

van der Poorten, P.M., Zhou, S. and Jones, C.B. (2002) "Topologically-consistent map generalisation procedures and multi-scale spatial databases", *Springer Lecture Notes in Computer Science 2478 (GIScience 2002)*, 209-227.

van Dijk, S.,Thierens, D. and de Berg, M.T. (1998) *Robust Genetic Algorithms for High Quality Map Labelling*, Utrecht University, UU-CS-1998-41.

Varanelli, J. and Cohoon, J.P. (1993) "Two-stage simulated annealing", *ACM Physical Design Workshop*, 50-53.

Varanelli, J. (1996) *On the acceleration of simulated annealing*, PhD Thesis, School of Engineering and Applied Sciences, University of Virginia, USA.

Visvalingam, M. and Whyatt, J. D. (1993) "Line generalisation by repeated elimination of points", *Cartographic Journal*, **30**(1), 46-51.

Wang, Z. and Müller, J-C. (1993) "Complex coastline generalisation", *Cartography and GIS*, **20**(2), 96-106.

Wang, Z. and Müller, J-C. (1998) "Line generalisation based on analysis of shape characteristics", *Cartography and GIS*, **25**(1), 3-15.

Ware, J.M. and Jones, C.B. (1992) "A multi-resolution topographic surface database", *International Journal of Geographical Information Systems*, **6**(6), 479-496.

Ware, J.M. and Jones, C.B. (1996) "A spatial model for detecting (and resolving) conflict caused by scale reduction", in *Proceedings of 7th International Symposium on Spatial Data Handling (Taylor & Francis)*, Delft, 547-558.

Ware, J.M. and Jones, C.B. (1998) "Conflict reduction in map generalisation using iterative improvement", *Geoinformatica*, **2**(4), 383-407.

Ware, J.M., Jones, C.B. and Thomas, N. (2003) "Automated cartographic map generalisation with multiple operators: a simulated annealing approach", *The International Journal of Geographical Information Science* (Taylor and Francis), **17**(8), 743-769.

Ware, J. M., Wilson, I. D., Ware, J. A. and Jones, C.B. (2002) "A tabu search approach to automated map generalisation", in *Proceedings of 10th ACM International Symposium on Advances in Geographical Information Systems (ACM)*, McLean VA, 101-106.

Ware, J.M., Wilson, I.D. and Ware, J.A. (2003) "A knowledge-based genetic algorithm approach to automating cartographic generalisation", *Knowledge-based Systems (Elsevier Science)*, **16**(5-6), 295-303.

Weibel, R. (1993) "Three essential building blocks for automated generalization", *GISDATA – Specialist Meeting on Cartographic Generalization*, Compiegne, France, Published in (Muller *et al.* 1995), 56-69.

Weibel, R. and Dutton, G. (1998) "Constraint-based automated map generalization", in *Proceedings of the 8th International Symposium on Spatial Data Handling (SDH)*, Vancouver, 214-224.

Weibel, R. and Jones, C.B. (1998) "Computational perspectives on map generalization", *Geoinformatica*, **2**(4), 307-314.

Wertheimer, M. (1912) Über das Denken der Naturvölker: I. Zahlen und Zahlgebilde. Zeitschrift für Pyschologie, **60**, 321-378.

Wilson, I.D., Ware, J.M. and Ware, J.A. (2003) "A genetic algorithm approach to cartographic map generalisation", *Computers in Industry (Elsevier Science)*, **52**(3), 291-304.

Yamamoto, M., Camara, G. and Lorena, L. (2002) "Tabu search heuristic for point-feature cartographic label placement", *Geoinformatica*, **6**(1), 77-90.

Zahn, Charles T. (1971) "Graph-theoretical methods for detecting and describing Gestalt clusters", *IEEE Transactions on Computers*, **20**(1), 68-86.

Zoraster, S. (1997) "Practical results using simulated annealing for point feature label placement", *Cartography and Geographical Information Systems*, **2**(4), 228-238.

INT. J. GEOGRAPHICAL INFORMATION SCIENCE, 2003
VOL. 17, NO. 8, 743–769

Taylor & Francis
Taylor & Francis Group

Research Article

Automated map generalization with multiple operators: a simulated annealing approach

J. MARK WARE

School of Computing, University of Glamorgan, Pontypridd CF37 1DL, UK; e-mail: jmware@glam.ac.uk

CHRISTOPHER B. JONES

Department of Computer Science, Cardiff University, Cardiff CF24 3XF, Wales, UK

and NATHAN THOMAS

School of Computing, University of Glamorgan, Pontypridd CF37 1DL, Wales, UK

(*Received 2 March 2002; accepted 10 April 2003*)

Abstract. This paper explores the use of the stochastic optimization technique of simulated annealing for map generalization. An algorithm is presented that performs operations of displacement, size exaggeration, deletion and size reduction of multiple map objects in order to resolve graphic conflict resulting from map scale reduction. It adopts a trial position approach in which each of n discrete polygonal objects is assigned k candidate trial positions that represent the original, displaced, size exaggerated, deleted and size reduced states of the object. This gives rise to a possible k^n distinct map configurations; the expectation is that some of these configurations will contain reduced levels of graphic conflict. Finding the configuration with least conflict by means of an exhaustive search is, however, not practical for realistic values of n and k. We show that evaluation of a subset of the configurations, using simulated annealing, can result in effective resolution of graphic conflict.

1. Introduction

The process of displaying map data at scales smaller than the source scale of the data can result in graphic conflicts in which map objects overlap or become too close together, or too small, to be clearly distinguishable. The cause of the conflicts is a combination of the effects of geometric scale reduction, whereby features decrease in size and increase in density on the map, and the fact that map symbols are often larger than the true-scale representation of their associated real-world feature. Thus with reduction in scale the map symbols may be regarded as competing for occupation of the limited space on the map display. The solution to such conflicts falls within the field of map generalization, whereby the content

International Journal of Geographical Information Science
ISSN 1365-8816 print/ISSN 1362-3087 online © 2003 Taylor & Francis Ltd
http://www.tandf.co.uk/journals
DOI: 10.1080/13658810310001596085

of a map is adapted to meets the constraints of the scale and the purpose of the map.

Typical operations performed in the course of map generalization include reduction in the detail of lines, areas and surfaces; deletion of less important features; exaggeration of the size of features that would otherwise be too small; amalgamation of neighbouring features; collapse in the dimensionality of area features to lines or points; caricature of the shape and pattern of simplified map features to communicate their original form; and displacement in order to retain adequate separation distances between neighbouring features. Solution of the specific problems of graphic conflict is usually met through the application of displacement, deletion and amalgamation operations, although all of the operations referred to have the potential to assist in the process. A further operation that may be employed in graphic conflict resolution is that of size reduction, though it is not usually referred to as a standard map generalization operation.

Various efforts have been made to automate all of the map generalization operations. Most of the early published procedures were developed to work on individual features, and hence ignore the problems of graphic conflict resolution. However, with increasing levels of use of map data in interactive and online displays, there is a particular need to develop procedures that resolve graphic conflicts fully automatically, without recourse to human intervention. Thus it may be envisaged that arbitrary combinations of digital map features, which approximate in level of detail to the users need, may be retrieved from a spatial database, but that their successful display requires online conflict resolution that must adapt to the user's *ad hoc* requirements.

Given a set of map features that is near to the required level of detail, the primary conflict resolution procedure may be regarded as displacement, since, when applied with care, this operator will not significantly reduce the information content of the map. By itself it cannot of course be guaranteed to resolve conflicts if there is insufficient map space available. Other operators such as deletion, amalgamation and size reduction may then be required if constraints of separation distances are to be met. There are several examples of the automation of displacement in local contexts, notably Nickerson (1988) for linear features, Mackaness (1994) for sets of points-referenced features and Jones *et al.* (1995) for area features.

A major challenge in automating displacement operations is to control the propagation of conflict that arises when the movement of an object may introduce a conflict with a feature that was not previously in conflict. This has given rise to a number of approaches that can process multiple map features. One approach is to formulate the problem as a set of mathematical constraints that can be solved analytically, giving a continuous solution space. Prime examples of this approach are Harrie (1999), who employed least squares methods, Højholt (2000) who applied finite element analysis, and Burghardt and Meier (1997) who made use of a method based on snakes. A second approach is to generate a discrete set of states that are searched as part of an iterative improvement procedure. Examples of discrete search methods are Ware and Jones (1998) who demonstrated the merits of simulated annealing to a set of predetermined candidate states, and Lonergan and Jones (2001) who applied a gradient descent procedure in combination with on the fly generation of displacement actions.

Each of the methods described previously have demonstrated good potential in

addressing the problem of conflict resolution, but none provides a method that can be guaranteed to resolve all graphical conflict, as they are based on application of displacement only. Harrie and Sarjakoski (2002) present a method that makes use of multiple operators to resolve graphic conflict that arises subsequent to scale reduction. The method, termed simultaneous graphic generalization, represents the problem of line displacement, simplification, smoothing and exaggeration as a set of linear constraints. Least squares adjustment is used to find a close to optimal solution to the constraints. The reactive displacement technique presented by Ruas (1998) is specifically aimed at building displacement. However, the method includes an initial density analysis stage to check whether or not displacement is appropriate; it has also been integrated into a more complete generalization tool (i.e. Stratège), which allows for initial building and road removal, and building aggregation based on density analysis.

The AGENT project (e.g. Lamy *et al.* 1999, Haire and Hardy 2001) advocates the use of intelligent software agents as a means of performing generalization. The idea is to represent individual map features as active agents, each having its own set of constraints that describe the way in which the feature should appear subsequent to generalization. In addition, an agent has the ability to measure the extent to which its constraints are met, and to suggest and try a range of conflict resolution strategies (e.g. generalization operations) until an optimal solution is found. Low level agents (e.g. buildings and road sections), are themselves under the control of higher level agents that control larger scale compound features (e.g. road networks and towns).

In this paper we show how simulated annealing (Kirkpatrick *et al.* 1983), an example of an iterative improvement technique, can be applied to multiple generalization operators of displacement, size exaggeration, deletion and size reduction to guarantee that constraints of proximity and size are met in the course of conflict resolution (figure 1). The methods presented use a trial position approach, which is similar to that used in point feature label placement (Zoraster 1997), and to the work of Ware and Jones (1998) on displacement. Each of n discrete polygonal objects is allocated k candidate trial positions into which it can possibly move; these positions represent the original, displaced, enlarged, displaced and enlarged, reduced, displaced and reduced, and deleted states of the object. This results in a possible k^n unique map configurations, some of which, it is assumed, will contain less conflict than the original. Identifying the best (or, at least, an

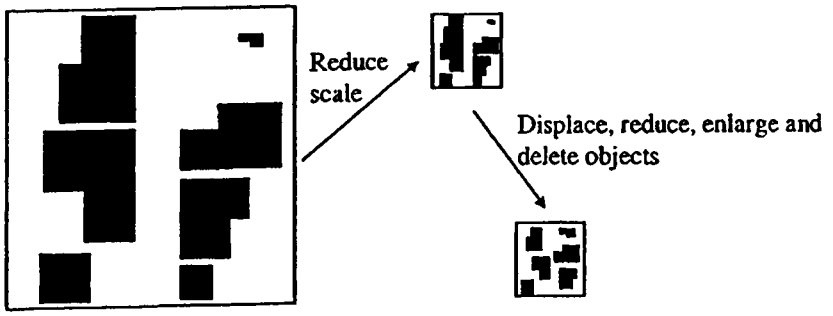


Figure 1. Reducing scale causes conflict, which can be resolved by a combination of object displacement, size enlargement, size reduction and deletion.

acceptable) configuration by means of an exhaustive search is, however, not practical for realistic values of n and k and hence an optimization technique is required. Simulated annealing search is a stochastic technique which has been shown to be successful in solving large optimization problems quickly (Emden-Weinert and Proksch, 1999), and we show here that it can be used successfully to reduce the number of map configurations needing to be generated and evaluated.

In §2 we summarise existing methods of applying simulated annealing to displacement of area objects, and highlight their shortcomings with regard to inadequate performance for online application, combined with failure to guarantee resolution of graphic conflict. In §3 we introduce techniques to improve significantly the performance of simulated annealing, based on partitioning and on a two-stage procedure, before introducing, in §4, the additional operators that when combined with displacement can guarantee to resolve graphic conflict. Experiments are described in §5 to show the importance of setting appropriate weights to control the application of the individual operations. The paper concludes in §6 with a summary of the results and a discussion of future work.

2. Conflict reduction by object displacement

In this section we present the problem of area object displacement in the context of fixed linear features, following the approach of Ware and Jones (1998). A map display D is regarded as made up of fixed linear objects F and n modifiable detached polygonal objects M. Each modifiable object $m_i \in M$ has k possible states, providing a total of k^n possible configurations of D.

2.1. *Object states*

At any given time a particular object m_i exists in one of its k states (we will also refer to these states as trial positions). An object's initial map position is designated as being trial position 1. Additional trial positions arise as a result of object displacement only. (Additional generalization operators are considered in §4). It is assumed that during the course of generalization, each object m_i can be displaced up to some maximum distance d from its original position (i.e. there is a continuous space that extends from m_i by a distance d into which it is permissible for m_i to move). The displacement trial positions associated with m_i represent a discrete approximation to this continuous space. Each object m_i has k displacement trial positions, which are distributed evenly about the object (figure 2).

2.2. *Evaluation of map display*

For a particular configuration D_j, each object m_i has an associated cost. The cost is determined by the extent to which m_i is in conflict. Two categories of spatial conflict are considered:

- PP-conflict. This involves conflict between a pair of polygonal objects (i.e. m_i and m_j lie too close to each other to be distinguishable). This conflict occurs when the minimum separating distance (in viewing coordinates) between m_i and m_j is less than some predefined threshold d_{minl}. An occurrence of this type of conflict carries a cost *ppcost*;
- PL-conflict. This involves conflict between a polygonal object and a linear object (i.e. m_i and $f_j \in F$ overlap or lie too close to each other to be

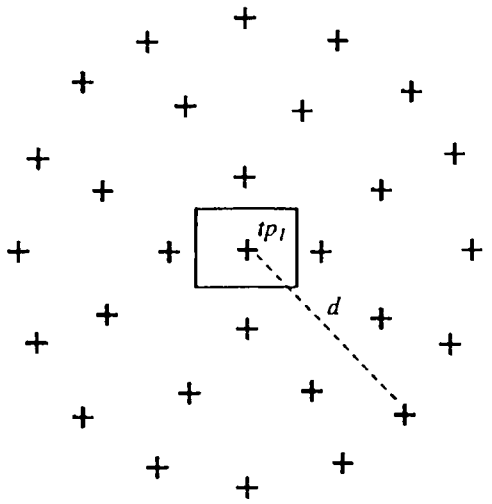


Figure 2. Displacement vector template for generating trial positions about a polygon. + = trial position, tp_1 = trial position 1. In this example, there are 28 displaced state trial positions.

distinguishable). This conflict occurs when the minimum separating distance (in viewing coordinates) between m_i and f_j is less than some predefined threshold d_{min2}. An occurrence of this type of conflict carries a cost *plcost*.

The total cost $C(D_j)$ associated with a realization D_j is found by summing the costs associated with each object $m_i \in M$. Our goal is to find a minimum cost configuration D_{min} such that:

$$C(D_{min}) = MIN(C(D_j), j = 1,2 \ldots k^n).$$

The set of all configurations is referred to as the search space. If the search space is small enough then D_{min} can be found by generating and evaluating each configuration D_j ($j = 1, 2 \ldots k^n$) in turn. However, this is not practical for realistic values of n and k. For example, a relatively simple display consisting of ten modifiable objects each with eight trial positions, gives rise to approximately 1 000 000 000 configurations.

2.3. *A simulated annealing solution*

A well-established approach to solving large optimization problems of the kind described is to adopt an iterative improvement algorithm. The concept of iterative improvement can be illustrated by considering the search space (i.e. in our case, all map configurations) to be laid out on the surface of a landscape. The elevation at any point on the landscape represents the cost for the particular configuration associated with that point. An iterative improvement algorithm will move around the landscape in an attempt to find the lowest troughs, which correspond to low cost configurations (Russel and Norvig 1995). Two major classes of iterative improvement algorithms are gradient descent and simulated annealing.

Algorithm 1 describes a simple gradient descent implementation. The algorithm accepts an initial map configuration $D_{initial}$ (i.e. each object in trial position 1), which is immediately designated as being the current solution $D_{current}$. Next, the

lowest cost successor D_{new} to $D_{current}$ is found. A particular successor to $D_{current}$ is found by moving a single object m_i to an alternative trial position; the lowest cost successor can be found by generating and evaluating all possible successors (of which there are $n(k\text{-}1)$). If D_{new} represents an improvement (in terms of cost) on $D_{current}$, then D_{new} becomes $D_{current}$ and the next lowest cost successor is generated. This process is repeated until a D_{new} is generated that offers no improvement; at this stage the algorithm terminates, with $D_{current}$ being returned as the solution. The algorithm is quite straightforward, but is not guaranteed to find an optimal solution since it is possible to arrive at a non-optimal current state from which no lower cost state can be reached. This occurs when the search descends into a local minimum, from which any single displacement generates a higher cost state. To use the landscape analogy once more, a local minimum can be thought of as a trough in the landscape that happens to be higher than the lowest point on the landscape. Several ways of trying to deal with the problem of local minima are available (e.g. random-restart, backtracking and multiple-moves). However, the exponential nature of most realistic search spaces makes such remedies impractical.

function **GradientDescent**
  input: $D_{initial}$
  $D_{current} \leftarrow D_{initial}$
  do
    $D_{new} \leftarrow$ **LowestCostSuccessor**($D_{current}$)
    if $\mathbf{C}(D_{new}) \geq \mathbf{C}(D_{current})$ then **Return**($D_{current}$)
    $D_{current} \leftarrow D_{new}$
  end
  Return($D_{current}$)

Algorithm 1. Gradient Descent.

Searches based on simulated annealing (Algorithm 2) attempt to overcome the problem of getting caught in local minima by sometimes allowing non-improving configurations to be accepted. They achieve this by taking advantage of the similarity between the way in which a metal cools from an initially high temperature until it freezes into a minimum energy crystalline structure (called the annealing process) and the search for a minimum in a more general system. As with gradient descent, simulated annealing always accepts D_{new} provided it offers a better solution than $D_{current}$. However, in cases where D_{new} provides no improvement, simulated annealing will accept the new configuration with some probability P ($P<1$). Like gradient descent, the algorithm begins by accepting an initial map configuration $D_{initial}$ (i.e. each object in trial position 1); this is immediately designated as being the current solution $D_{current}$. Next a random successor D_{new} is generated by moving a randomly chosen object m_i to a randomly chosen trial positions k_j. If the displacement results in a display configuration with a lower cost ($\mathbf{C}(D_{new}) < \mathbf{C}(D_{current})$), then the object remains in the chosen trial position ($D_{current} \leftarrow D_{new}$). If, however, the new display has a higher or equal cost (i.e. $\mathbf{C}(D_{new}) \geq \mathbf{C}(D_{current})$), then the object is either returned to its previous position or remains in its new position, depending on probability P. The process of attempting a random object displacement continues until stop conditions are met (e.g. a solution that meets a target cost is found or a pre-defined maximum number of iterations have taken place or a pre-defined maximum amount of time has elapsed).

function **SimulatedAnnealing**
  input: $D_{initial}$, *Schedule*, *Stop_Conditions*
  $D_{current} \leftarrow D_{initial}$
  $T \leftarrow$ **initialT**(*Schedule*)
  while **NotMet**(*StopConditions*)
    $D_{new} \leftarrow$ **RandomSuccessor**($D_{current}$)
    $\Delta E \leftarrow$ **C**($D_{current}$)-**C**(D_{new})
    if $\Delta E > 0$ then $D_{current} \leftarrow D_{new}$
    else
      $P = e^{-\Delta E/T}$
      $R =$ **Random**(0,1)
      if ($R < P$) then $D_{current} \leftarrow D_{new}$
    end
    $T \leftarrow$ **UpdateT**(*Schedule*)
  end
  Return($D_{current}$)

Algorithm 2. Simulated Annealing.

At each iteration the probability P is dependant on two variables: ΔE (the change in conflict, measured by the difference in cost between the new and current states); and T (the current 'temperature'). P is defined as:

$$P = e^{-\Delta E/T} \tag{1}$$

T is assigned a relatively high initial value; its value is decreased in stages throughout the running of the algorithm. At high values of T poor displacements (large negative ΔE) will often be accepted. At low values of T poor displacements will tend to be rejected (although displacements resulting in small negative ΔE might still sometimes be accepted). The acceptance of some poor displacements is permitted so as to allow escape from locally optimal solutions. In practice, the probability P is usually tested against a random number R ($0 \leq R \leq 1$). A value of $R < P$ results in the new state being accepted. For example, if $P = 1/3$, then we would expect, on average, for every third worse new state to be accepted. The initial value of T and the rate by which it decreases is governed by what is called the annealing schedule. Generally, the higher the initial value of T and the slower the rate of change, the better the result (in cost reduction terms); however, the processing overheads associated with the algorithm will increase as the rate of change in T becomes more gradual.

Finding a minimum cost configuration D_{min} by simulated annealing is statistically guaranteed, provided that the reductions in T are small enough and that for each value of T the number of configurations tested is large enough (Zoraster 1997). However, most practical applications settle for near optimal solutions, and make corresponding compromises in the annealing schedule; a suitable schedule is usually decided upon after some preliminary experimentation.

2.4. *Cost function*

The viability of any iterative improvement algorithm depends heavily on it having an efficient cost function, the purpose of which is to determine for any given element of the search space (i.e. any map realization) a value that represents the relative quality of that element. The cost function used here, C, is called repeatedly

and works by calculating and summing the costs associated with objects $m_i \in M$. When invoked initially, C must calculate the cost associated with every object $m_i \in M$. A record of these costs is maintained for future reference, meaning that, in any further call, C has to consider only objects with costs affected by the most recent displacement. Calculating the cost associated with a polygonal object m_i involves identifying all other polygonal objects lying within a distance d_{min1} and all linear objects lying within a distance d_{min2}. Identifying these conflicting objects quickly requires the use of a spatial index of some kind. The current implementation makes use a triangle-based data structure, together with an associated search procedure (Jones and Ware 1998).

2.5. *Initial results*

Initial experiments reported by Ware and Jones (1998) demonstrated that both gradient descent and simulated annealing approaches are successful in reducing graphic conflict while limiting the number of realizations examined. When compared against each other, the simulated annealing approach was clearly superior with regard to the degree of conflict reduction achieved. The experiments reported made use of IGN-France BDTopo data (1:25 000) consisting of 321 polygonal objects contained within 16 regions. In the experiments, PP-conflict is deemed less serious than PL-conflict; the *ppcost* and *plcost* are set so as to reflect this fact. The simulated annealing algorithm was implemented originally in C under UNIX on a Sun Enterprise 2 model 2200 (2×200 MHz Ultrasparc).

A key factor determining the success or failure of simulated annealing is the choice of annealing schedule and the one used was based on the format of Christensen *et al.* (1995). This involves setting T to an initial value τ. At each temperature a maximum of ωn object displacements (successful or unsuccessful) are allowed. After every ωn displacements T is decreased such that $T_{new} = T_{old} - \lambda\tau$. Also, if more than ζn successful displacements are made at any one temperature then T is immediately decreased. If no successful displacements are made at a particular temperature then the algorithm terminates. Finally, a limit on the maximum number of temperature stages allowed is set to ψ (in practice the maximum number of temperature stages is never reached). The annealing parameters used were: $\tau = 3.0$, $\lambda = 0.1$, $\omega = 100$, $\zeta = 30$ and $\psi = 50$. These values were arrived at via experimentation. The other parameters used were: $d_{min1} = 7.5$ m, $d_{min2} = 7.5$ m, $d = 7.5$ m, $ppcost = 1$ and $plcost = 10$. Note that the minimum separating distance tolerance values used (together with the tolerance values used in §5.2–5.5) were chosen so as to provide initial conflict sufficient to demonstrate the usefulness of the approach. Tolerance values that reflect a specific cartographic specification are used in the experiments presented in §5.6. Prior to annealing, there were 236 occurrences of PP-conflict and 36 occurrences of PL-conflict. As can be seen from the results given in table 1 and illustrated in figure 3, the simulated annealing approach reduces the amount of graphic conflict by up to 90%. It achieves this while at the same time limiting the number of realizations generated and evaluated (approximately 340 000 out of a possible 29^{321}).

2.6 *Updated results for displacement-only conflict resolution*

For the sake of valid comparison with results reported in this paper, the C implementation referred to above has been run under LINUX on an 800 MHz

Table 1. Original and updated results.

Results	average PP-conflict	s.d. PP-conflict	average PL-conflict	s.d. PL-conflict	average number of tests	s.d. number of tests	average execution time (s)	s.d. execution time (s)
Original	26.6	2.9	0.0	0.0	342302.2	18613.1	39.67	2.17
800$_{old}$	25.4	2.7	0.0	0.0	328282.5	17439.3	13.46	0.67
800$_{new}$	20.8	1.1	0.0	0.0	302840.0	23363.7	11.83	1.12

Pentium III machine (128 Mb RAM). The results are shown in table 1 (see 800$_{old}$). The only significant change is in execution time, which has been reduced from 39.67 s to 13.46 s; this reduction is a direct result of using a faster machine. The number of realizations tested and final conflict remain more or less the same; the small differences are due to the random nature of simulated annealing.

Our implementation of the displacement only procedure makes two minor modifications to the original implementation. The first change concerns the setting of the initial temperature τ. Previously this value was supplied as a user defined input parameter (a suitable value being arrived at via experimentation). A more flexible approach is adopted here, in which τ is generated automatically. The method used is based on that described by Zoraster (1997). For the first 500 iterations (the value 500 was arrived at via experimentation), the algorithm accepts non-improving moves with probability 1/3. At this point the mean value $\mathbf{m}(\Delta E)$ for non-improving moves encountered (during the 500 iterations) is calculated. τ is then computed so that P is 1/3 for the average value of $\mathbf{m}(\Delta E)$:

$$\tau = \mathbf{m}(\Delta E)/(\ln 3).$$

The second change concerns the annealing 'cooling' schedule, and in particular the rate at which T is decreased. Instead of T being decreased linearly (i.e. $T_{new} = T_{old} - \lambda\tau$), it is reduced geometrically, such that $T_{new} = \lambda T_{old}$. Experiments show the modified implementation to work best with the following annealing parameters: τ = auto, $\lambda = 0.9$, $\omega = 40$, $\zeta = 20$ and $\psi = 50$. The results presented in table 1 (see 800$_{new}$) indicate improvements, both in terms of graphic conflict removal and execution time.

3. Execution time improvement

A shortcoming of the use of simulated annealing to date is that execution times are slow, particularly if it is to be considered for use in applications requiring on-the-fly generalization (e.g. web mapping, in-car navigation and location based services). This section describes two techniques for reducing execution times. Both techniques achieve improvement by reducing the number of realizations having to be generated and tested.

3.1. *Data partitioning*

Firstly, we suggest that execution can be speeded up by dividing the map into autonomous regions (i.e. such that there is no possibility ever of objects in a particular region coming into conflict with objects in any other region). The reasoning here is that by dividing the data, the annealing process (specifically τ and the rate of cooling) can respond to meet more appropriately the specific demands of each region. One problem to be solved here is finding a technique for dividing up

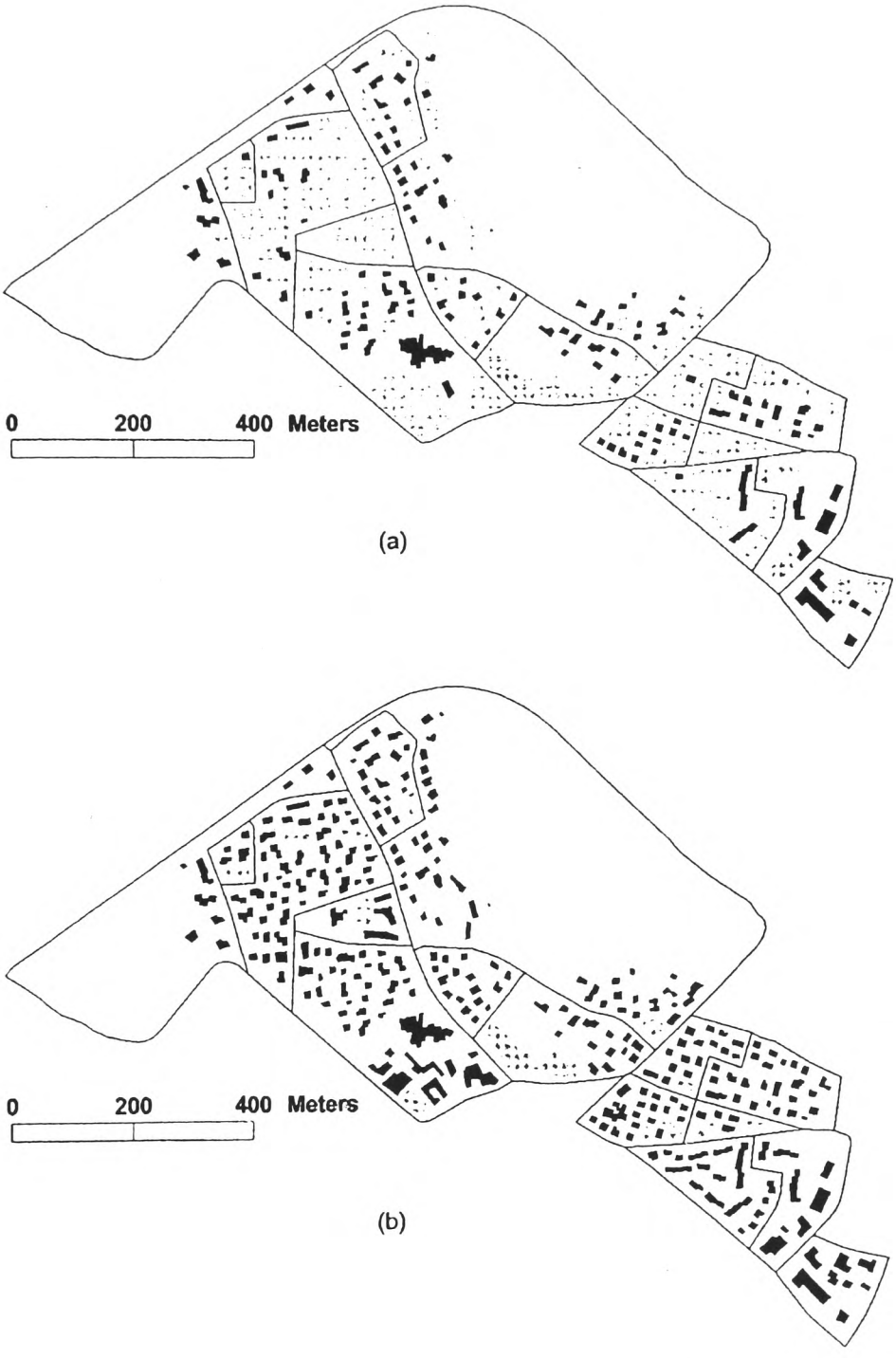


Figure 3. (a) Original BDTopo data. (b) BDTopo data after application of simulated annealing algorithm. Objects shown in grey are involved in spatial conflict of some kind.

the map. In some instances it may be possible to make use of naturally occurring regions, such as those formed by a road network or administrative boundaries. Other situations will require analysis of the distribution of objects in order to find groupings of objects that sustain no influence from objects external to their group. We make use of a road network to divide data; the technique was used previously by Ruas (1998). Using this approach, the BDTopo is divided into 16 regions (figure 4). The simulated annealing algorithm can now be applied to regions individually. In each case the annealing schedule used is the same as that used previously (with initial temperature τ set automatically). The results obtained are given in table 2. It can be seen that, on average, the total number of realizations tested has been reduced to 236935.6. The average execution time sees a corresponding fall to 9.73 s, a reduction of approximately 18%. The conflict remaining in the data has not changed significantly.

3.2. *Two-stage approach*

Some authors suggest that optimization can be made more efficient by adopting a two-stage approach (Varanelli and Cohoon 1995). In two-stage simulated annealing a faster heuristic algorithm is used to replace the simulated annealing actions occurring at higher temperatures. This is followed by a conventional simulated annealing approach initiated at lower temperatures in an attempt to improve on the initial solution. The approach adopted in our implementation differs slightly in that the faster heuristic algorithm is replaced by simulated annealing in conjunction with an annealing schedule that involves an initial high

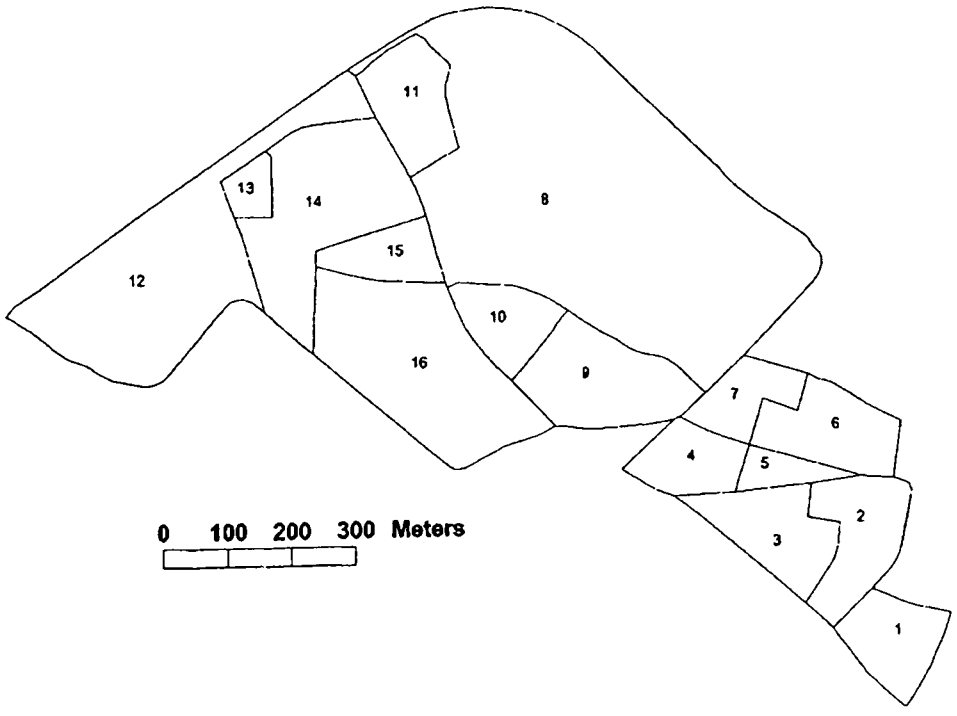


Figure 4. Division of BDTopo data into 16 autonomous regions.

Table 2. Data partitioning results.

Region	Initial		Results (averages)			
	PP-conflict	PL-conflict	PP-conflict	PL-conflict	number of tests	execution time (s)
1	2	1	0	0.0	94.8	0.0038
2	4	0	0	0.0	96.2	0.0039
3	16	1	0.8	0.0	17562.6	0.6849
4	6	7	0.4	0.0	21185.8	0.8199
5	8	1	2	0.0	10800.0	0.4644
6	12	2	0	0.0	9292.6	0.3531
7	24	0	2.4	0.0	21128.0	0.8731
8	18	4	0	0.0	21386.8	0.8747
9	10	7	2	0.0	16448.0	0.6558
10	6	0	0	0.0	4600.4	0.1895
11	12	2	2.4	0.0	18114.4	0.7046
12	0	1	0	0.0	5.4	0.0002
13	6	1	1.2	0.0	8189.2	0.3521
14	70	1	4.8	0.0	47488.0	2.1512
15	6	2	2	0.0	8624.0	0.3525
16	36	6	3.2	0.0	31919.4	1.2448
Total	236	36	21.2	0.0	236935.6	9.7285

temperature followed by rapid cooling. The second stage again makes use of simulated annealing, this time with a much lower initial temperature followed by much slower cooling. The annealing parameters used are given in table 3. For the second stage, experiments showed that more consistent results were obtained by reverting to a fixed value of τ. Experimental results (table 4) show that execution times are reduced by just less than 50% when adopting the two-stage approach.

Table 3. Parameters used in two-stage annealing.

Stage	τ	λ	ω	ζ	ψ
1	auto	0.6	20	10	50
2	0.25	0.9	50	25	50

Table 4. Two-stage approach results and combined results (previous results also shown).

Results	average PP-conflict	average PL-conflict	average number of tests	average execution time (s)
Original	26.6	0.0	342302.2	39.67
800_{old}	25.4	0.0	328282.5	13.46
800_{new}	20.8	0.0	302840.0	11.83
Partitioned	21.2	0.0	236935.6	9.73
Two-stage	22.4	0.0	150749.2	6.13
Combined	21.5	0.0	102539.4	4.1

3.3. *Overall improvement*

The two modifications described have been combined in a single implementation (i.e. the two-stage procedure is applied to each of the 16 regions in turn). Experimental results, shown in table 4, reveal an overall improvement of approximately 65% with average execution times falling from 11.83 s to 4.1 s.

3.4. *Displacement cost*

Objects are displaced during the course of annealing, the overall aim being to reduce graphic conflict. However, many of the displacements prove unnecessary (i.e. they do not lead ultimately to a reduction in graphic conflict) and occur only as a consequence of the algorithms occasional acceptance of neutral and negative displacement. The result is a final display containing objects displaced from their original location without benefit. In order to minimize displacement of this type, an object displacement cost is introduced. If an object exists in a displaced state then a cost is incurred:

- Displacing an object carries a cost $\delta dispcost$, where δ represents the magnitude of displacement.

The costs associated with graphic conflict and object displacement are combined to give the overall cost associated with an object. For example, consider an object m_i that currently exists in a displaced state (cost $=\delta dispcost$) and lies in conflict with two other polygonal objects (cost $=2ppcost$) and one linear object (cost $=plcost$). Its associated cost would equal ($2ppcost+plcost+\delta dispcost$). Assigning appropriate values to *ppcost*, *plcost*, and *dispcost* (i.e. $0<dispcost<ppcost$, *plcost*) creates an incentive for displaced objects to return, during the course of annealing, as near to their original location as is possible provided there is no resulting increase in graphic conflict. Displays produced with and without consideration to displacement cost are shown in figure 5.

4. Additional operators

A further shortcoming of the initial algorithm is that it does not guarantee the removal of all graphic conflict. For example, the best result obtained during experiments was a final PP-conflict cost of 18. It is clear that displacement on its own is not sufficient to resolve all graphic conflict and additional generalization operators are required. We consider three additional operators:

- size exaggeration;
- deletion;
- size reduction.

Size exaggeration is required in situations where an object becomes too small to be viewed adequately at the target scale. In situations where there is not enough space for all objects to displayed, some objects have to be deleted. An alternative to deletion is to reduce slightly the size of relatively large objects.

4.1. *Object states*

Again we consider a map display D made up of fixed linear objects F and n modifiable detached polygonal objects M. Each modifiable object $m_i \in M$ has k

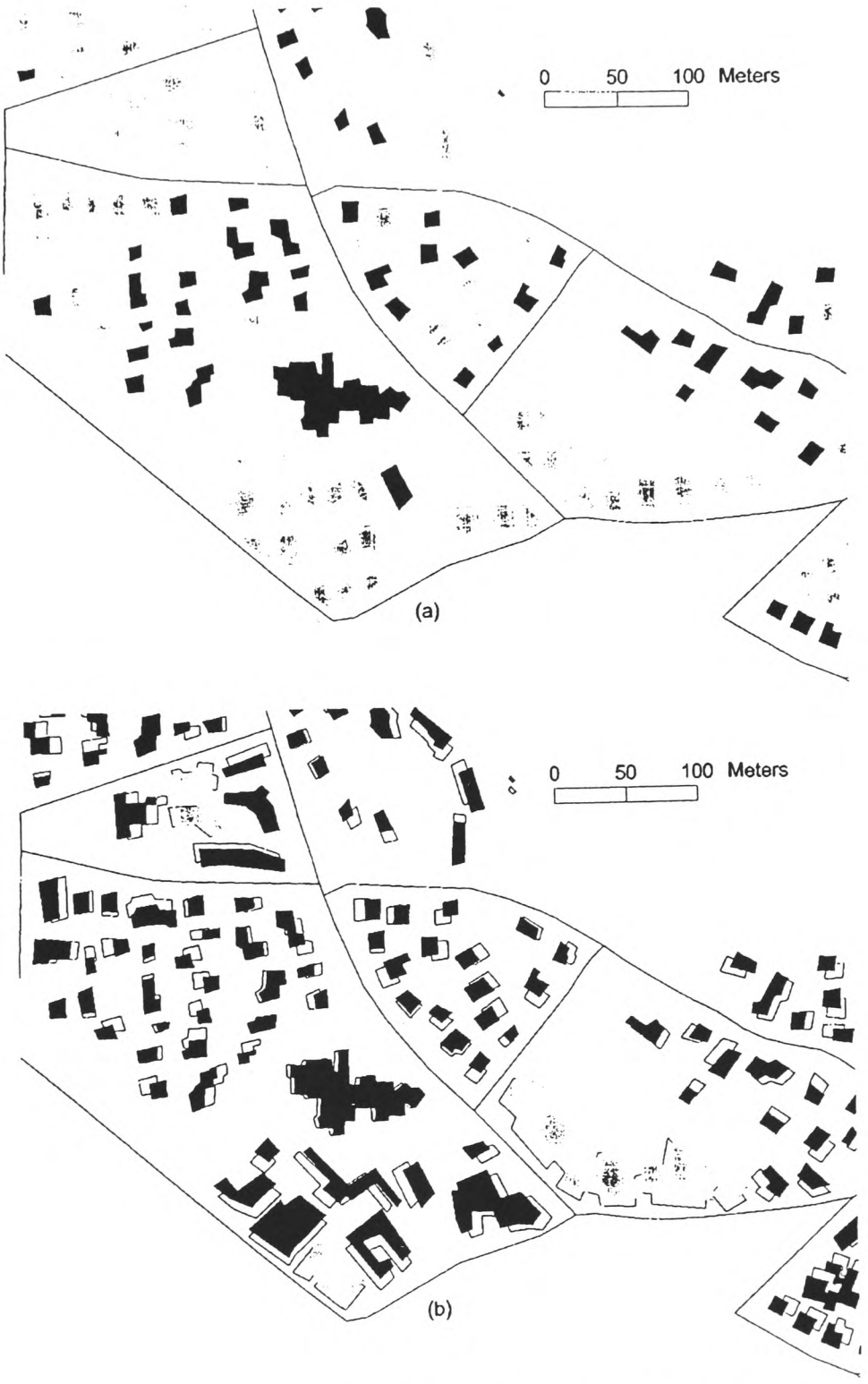


Figure 5(a–b). Example of the effect of including displacement cost. (a) Original data. (b) Result obtained without consideration to displacement cost (original object locations shown in background).

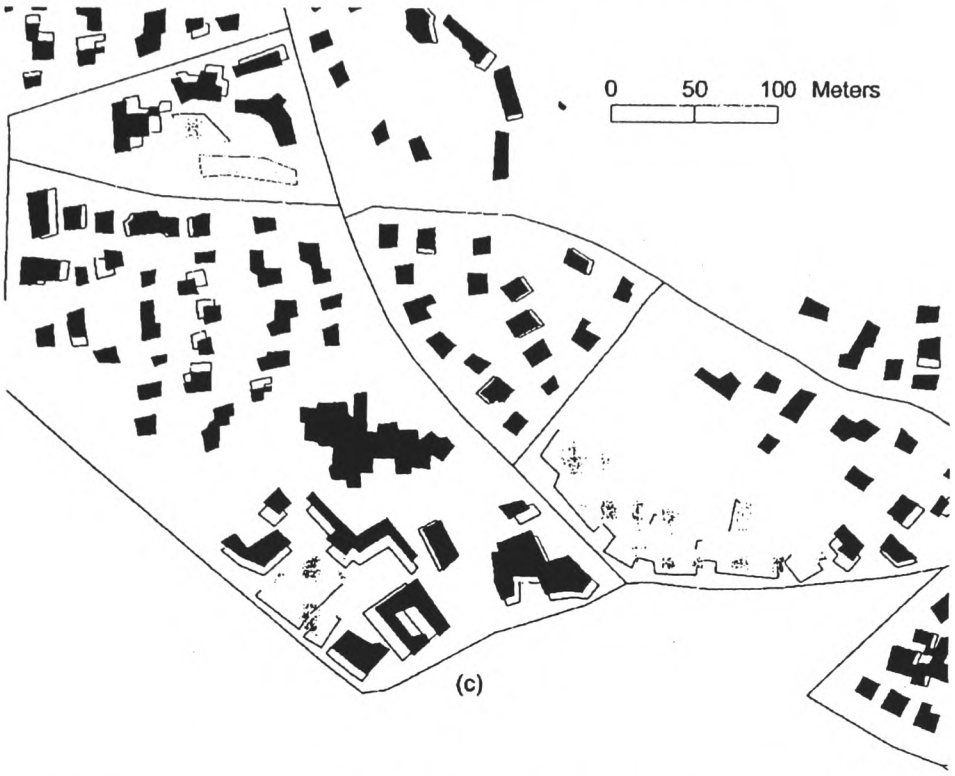


Figure 5(c). Example of the effect of including displacement cost. (c) Result obtained when displacement cost is taken into account; unnecessary displacement has been reduced.

possible states providing us with k^n possible configurations of D. At any given time a particular object m_i exists in one of its k possible states. An object's (k-1) modified states arise as a result of displacing the object, exaggerating the size of the object, reducing the size of the object, displacing and exaggerating the object, displacing and reducing the object, or deleting the object. In the current implementation the possible object states are:

- *Unmodified State.* Each object m_i has a single unmodified state;
- *Displaced States.* Each object m_i has q displaced state trial positions that are distributed evenly about the object;
- *Enlarged States.* Each object m_i has $q+1$ enlarged state trial position; these trial positions result from applying a scaling factor s_e ($s_e \geq 1.0$) to the unmodified state and displaced states of m_i;
- *Deleted State.* An object m_i has a single deleted state trial position; this trial position represents the situation where the object has been removed from the display;
- *Reduced States.* Each object m_i has $q+1$ reduced state trial positions; these trial positions result from applying a scaling factor s_r ($0.0 < s_r < 1.0$) to the unmodified state and displaced states of m_i.

There are therefore a total of ($3q+4=k$) trial positions. The object reduction factor s_r is provided as an input parameter to the algorithm. The object enlargement

value s_e varies for each object and is dependant on object display area. For an object with display area less than a minimum area tolerance a_{min}, s_e is set so as to increase object display area to a_{min}. Objects with display area greater than or equal to a_{min} have s_e set to 1.0 (i.e. its application has no effect).

4.2. *Evaluation of map display*

For a particular configuration D_j, an object m_i has an associated cost. This cost is a measure of both the graphic conflict in which the object is involved and the extent to which the object is modified. There are now three categories of graphic conflict. The first two are the PP-conflict and PP-conflict types described previously. The third is as follows:

- PA-conflict. Conflict involving a single polygonal object (i.e. m_i is too small for it to be seen clearly). This type of conflict occurs when the area of m_i (in viewing coordinates) is less than a_{min}. An occurrence of this type of conflict carries a cost *pacost*.

If an object exists in a modified state then a cost is incurred:

- Displacing an object carries a cost δ*dispcost*, where δ represents the magnitude of displacement from the original position;
- Enlarging an object carries a cost *ecost* that is proportional to the scale of enlargement, relative to the original size;
- Deleting an object carries a cost *delcost*;
- Reducing an object carries a cost *rcost* that is proportional to the scale of reduction, relative to the original size.

The costs associated with graphic conflict and object modifications are combined to give the overall cost associated with an object. For example, consider an object m_i that has been reduced in size and lies in conflict with two other polygonal objects and one linear object. Its associated cost would equal (2*ppcost*+*plcost*+*rcost*). As before, the total cost $C(D_j)$ associated with a realization D_j is found by summing the costs associated with each object m_i; our goal is to find a minimum cost configuration D_{min}.

5. Implementation and testing

In implementation terms, introducing size enlargement, size reduction and deletion capabilities is straightforward; these additional modified states of an object are treated as additional trial positions for that object. Size reduction and size enlargement are achieved by applying a suitable scaling factor (s_r and s_e) to the object, while deletion is accommodated by means of a simple Boolean flag (i.e. an object is either deleted or not deleted). The cost function **C** is updated to take account of the additional costs.

5.1. *Cost setting*

It is important to make sure that conflict costs (*ppcost*, *plcost* and *pacost*) and object modification costs (*dispcost*, *ecost*, *rcost* and *delcost*) are set appropriately; it is these costs that govern the likelihood of any given object/trial position pairing

being accepted should they be chosen during the annealing process. As such, the cost values must be set so as to accommodate any orders of precedence that might exist between the various operators and conflict types. For example, consider an application that simply requires the removal of all graphic conflict (i.e. reducing graphic conflict is the stated primary goal, and minimizing object modification is an implied secondary goal). This might be achieved by assigning a relatively high value to each of the conflict costs (e.g. $ppcost = 100$, $plcost = 100$ and $pacost = 100$) and a relatively low value to each of the object modification costs (e.g. $dispcost = 5$, $ecost = 5$, $rcost = 5$ and $delcost = 5$). It could be the case, however, that the modification operators have an order of precedence. For example, object displacement might be preferred to size reduction, which in turn might be preferred to deletion; the relevant cost value are changed accordingly (e.g. $dispcost = 5$, $rcost = 10$ and $delcost = 15$).

5.2. *Displacement and deletion*

Deletion was the first of the additional operators to be tested. The main thing to note here is that care must be taken when setting the *delcost* value. If it is set too high then, in some situations, not enough deletion takes place, and, as a consequence, graphic conflict is not always removed (i.e. the cost of deleting is greater than the cost associated with the graphic conflict). On the other hand, if *delcost* is set too low, then too many deletions tend to occur (i.e. objects are prematurely deleted in situations where displacement could have succeeded). An example of each of the *delcost* setting scenarios is given in figure 6.

5.3. *Reduction and enlargement*

Object size enlargement and object size reduction operators are now introduced. Again note that care must be taken when setting object modification costs and regard must be given to any operator precedence that may exist. Figure 7 shows a display produced with object displacement preferred to size reduction, and size reduction preferred to deletion.

5.4. *Cost weighting*

When generalizing a map it is important to consider the relative importance of objects; important objects should be less prone to modification than unimportant objects. Consider a tourist map in which an object m_m, representing a museum, lies in conflict with an object m_h, representing an ordinary house. In this context, m_m can be regarded as being more important than m_h, and hence should be less susceptible to generalization. In this situation there are a number of alternative conflict resolution strategies that could be employed. For example, conflict resolution could initially involve displacement and reduction of m_h only; if this failed, m_h could be deleted. An alternative strategy might again involve the initial displacement and reduction of m_h only. If this failed then the next step would be displace and reduce m_h and m_m in combination; continued failure at this stage would result in the deletion of m_h. Relative object importance is incorporated into the simulated annealing procedure by assigning a cost weighting w_i to each object m_i. Whenever an object m_i and trial position k_j pairing are chosen during the annealing process, the cost associated with k_j is multiplied by w_i to give a modified

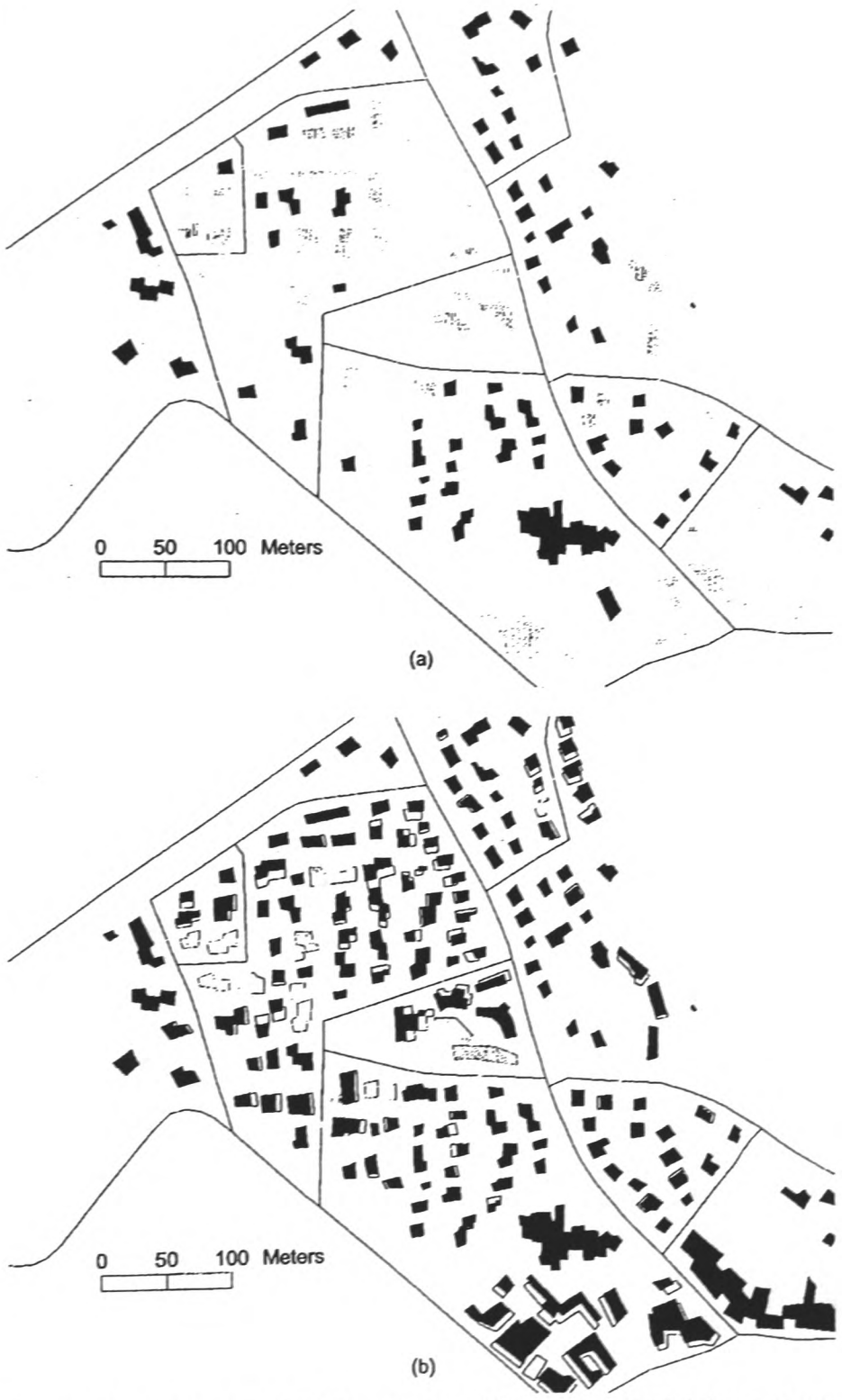


Figure 6(a–b). Application of deletion operator. (a) Original data. (b) Deletion cost set too high, not all spatial conflict removed.

Figure 6(c–d). Application of deletion operator. (c) Deletion cost set too low, over-deletion. (d) A more appropriate deletion cost setting.

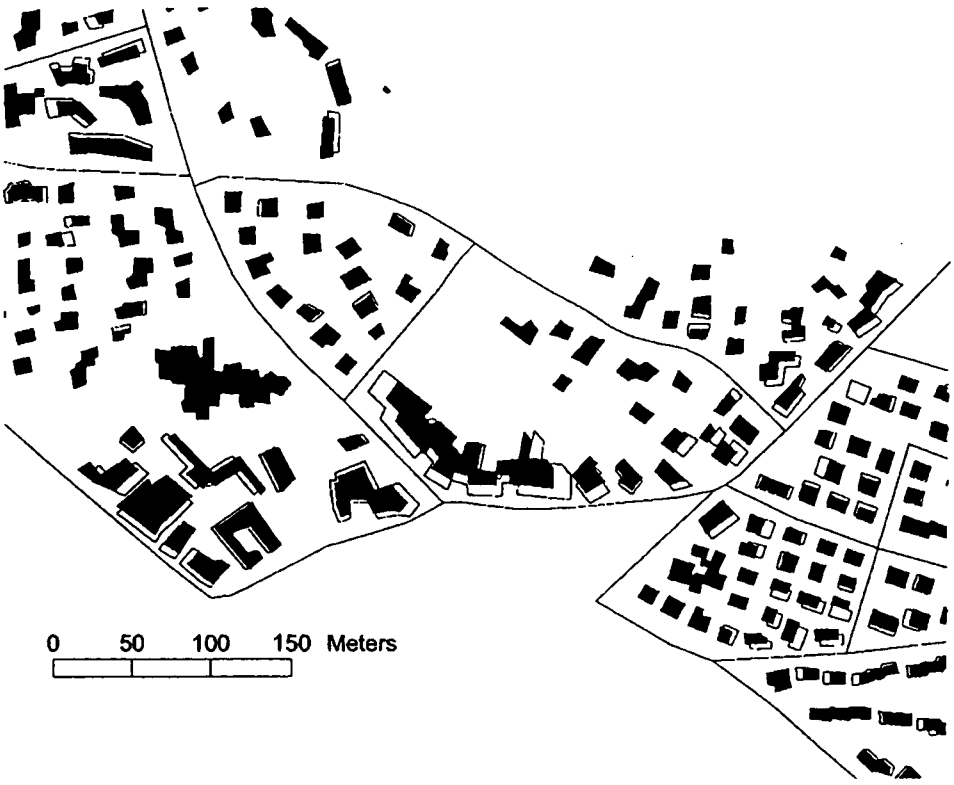


Figure 7. Result obtained using displacement, size enlargement, size reduction and deletion.

cost. A particular cost weighting value w_i will be based on one or more attributes of m_i. In our experiments to date, and in the absence of any other measure of importance, an object's importance, and hence its cost weighting value, is assumed to be proportional to object area (with large objects deemed more important than small objects). Figure 8(a) shows output produced with no account taken of object importance. The large object to the left has been deleted in order to resolve conflict. If we consider large objects to be more important than small, then it would make more sense to have deleted the smaller object to the right. This can be achieved by making use of appropriate cost weightings, as shown in figure 8(b).

5.5. *Performance*

Initial experiments using the complete set of operators were shown to work well with the following parameters: first-pass (τ=auto, $\lambda=0.6$, $\omega=20$, $\zeta=10$ and $\psi=50$); second-pass ($\tau=5.0$, $\lambda=0.9$, $\omega=40$, $\zeta=20$ and $\psi=50$). The example output shown in figure 7 made use of the following costs: $ppcost=5$; $plcost=50$; $pacost=1.5$; $dispcost=0.1$, $ecost=0.5$, $rcost=0.5$ and $delcost=2.5$. The other values used were: $d_{min1}=7.5$ m, $d_{min2}=7.5$ m; $a_{min}=40\,m^2$; $d=7.5$ m; and $s_r=0.8$. The average number of realizations tested was approximately 321421.7, with average execution times of 12.75 s. All PP-conflict and PL-conflict was removed. On average, 4.2 objects were deleted, 24.8 objects were size reduced and 114.3 objects displaced.

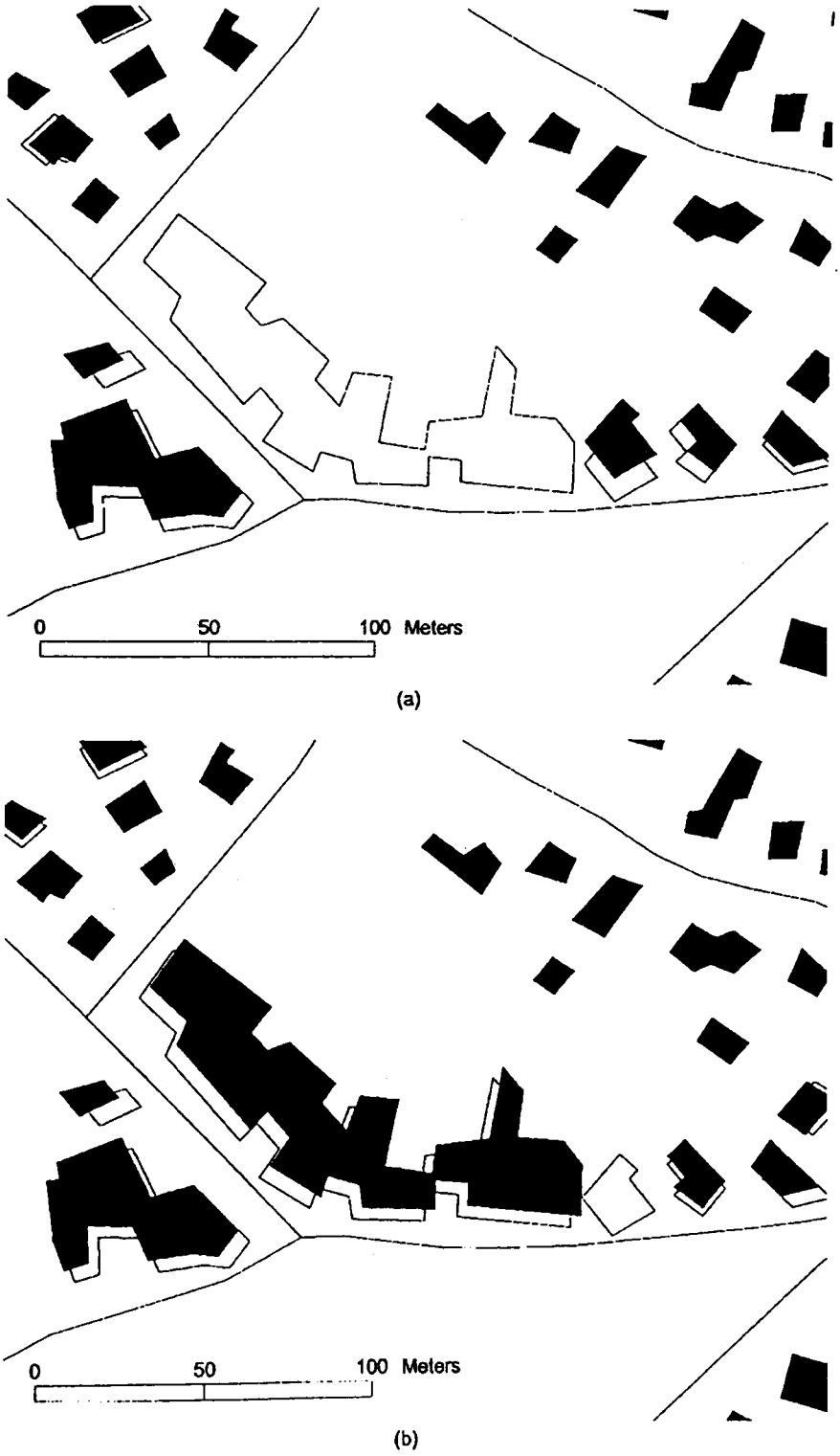


Figure 8. The effect of cost weighting. (a) Result obtained without weighting – large object deleted. (b) Result obtained with area weighting – small object deleted.

5.6. *Application to large scale ordnance survey data*

The simulated annealing application has also been tested on large scale (1:1250) Ordnance Survey (OS) data. The data consists of Mastermap building polygons (pre-processed by OS staff using ArcInfo generalization tools) and OSCAR road centre-lines. The tests assumed a map scale reduction to 1:10 000, with the road centre-lines being assigned a symbol width of 14.0 m (e.g. figures 9(*a*) and 10(*a*)). The annealing parameters and conflict cost values used were the same as those given in §5.5. The other values used were: $d_{min1} = 1.0$ m, $d_{min2} = 7.0$ m (i.e. 14.0m/2); $a_{min} = 40\,\text{m}^2$; $d = 10.0$ m; and $s_r = 0.9$. Sample output, showing examples of displacement, size enlargement and size reduction, is shown in figures 9(*b*) and 10(*b*).

6. Conclusion

This paper has presented a simulated annealing based algorithm for map generalization. The algorithm carries out map conflict reduction using displacement, size enlargement, deletion and size reduction operators in combination. Experimental results have shown the algorithm to be successful in reducing graphic conflict within reasonable time. In more general terms, the work has further highlighted the potential of iterative improvement techniques to act as a means of orchestrating parts of the map generalization process.

A possible shortcoming of the existing approach is that it makes use of only a fixed number of pre-defined trial positions; in some conflict situations the set of available trial positions might not provide an appropriate solution. For example, figure 9 shows how the algorithm can sometimes result in excessive displacement and excessive reduction of building features. It may be possible to overcome this problem by increasing the resolution of the search space by adding trial positions. In the case of size reduction, the number of reduced states could be increased by making use of a range of scaling values. This approach will be investigated as part of the ongoing program of research. An alternative solution, which will also be investigated, might be to adopt a continuous search space, as advocated by Strijk and van Kreveld (2002) for the purpose of point labelling. The idea here would be to generate random trial positions (i.e. randomly chosen operators together with the appropriate randomly generated operator parameters) on the fly.

Future research will also focus on the development of additional operators, such as simplification and aggregation. Conceptually this is seen as relatively straightforward; each additional operator will produce additional trial positions (either fixed or random) for any object to which it might be supplied. It is also the intention to include operators that act upon line, point and textual data (i.e. line, point and text objects will have associated operators and corresponding trial positions).

In addition, the cost function will be expanded to take disruption to the structure, form and density of map features into account. For example, displacing a building might result in a row of buildings become misaligned (see figure 9 for example). Attaching a high cost to such misalignment will stop the displacement from taking place in the first place or encourage future displacements to realign the buildings or stimulate some other remedial course of action (e.g. delete the offending building).

In the current implementation, modification operators are applied to a single

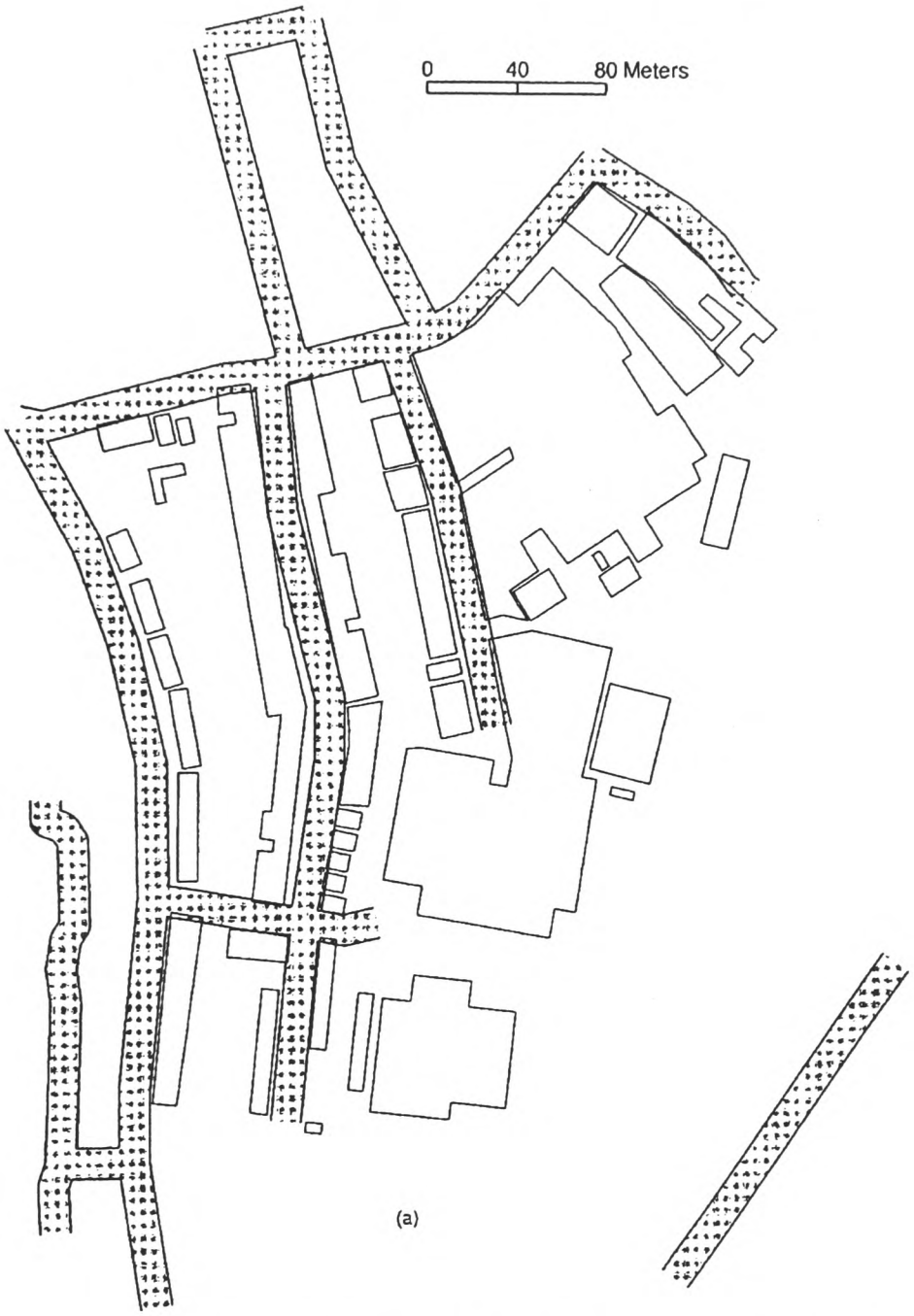


Figure 9(a). Large scale OS data example. (a) Graphic conflict arising due to scale reduction and road symbolization.

object only at any given time. Other strategies could be accommodated. For example, there is no reason why deletion could not be applied to groups of objects en-mass, where groupings are determined by some attribute such as object class or

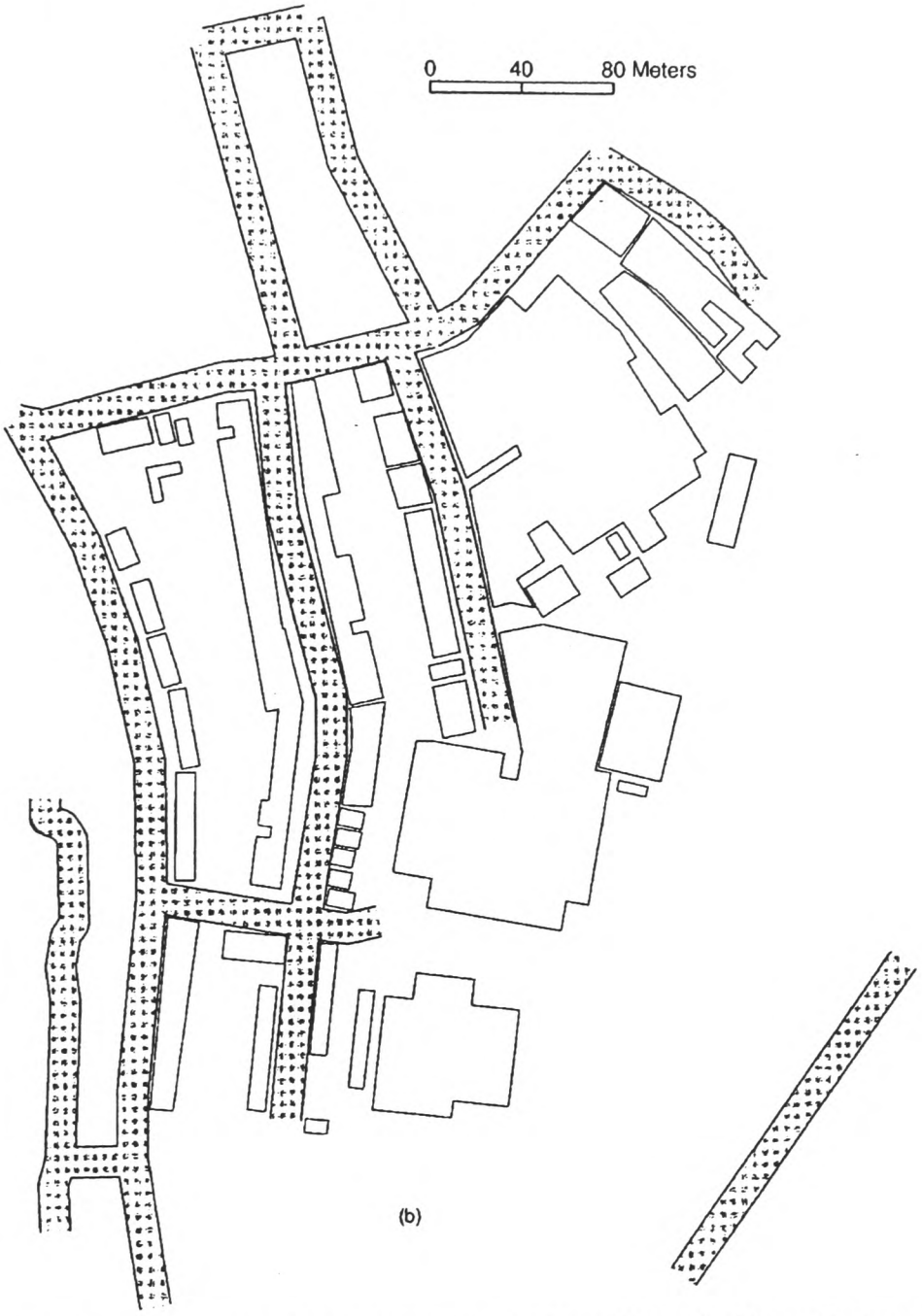


Figure 9(b). Large scale OS data example. (b) Graphic conflict resolved by application of displacement, size enlargement, size reduction and deletion.

object area. The en-mass deletion could be treated as just another trial position for the objects in question (i.e. the objects could be reintroduced as a group at some later stage of the annealing process), or could be applied as a more permanent

Figure 10(a). Large scale OS data example. (a) Graphic conflict arising due to scale reduction and road symbolization.

culling of objects at pertinent stages of the generalization process (e.g. at the start following an initial assessment of the problem or at certain stages during the generalization process when it becomes apparent that the other operators will not succeed in resolving conflict). Similarly an associated collection of objects, perhaps representing a row of buildings, could be displaced (or enlarged or reduced) as a group, so as to maintain, for example, feature alignment.

Another area for future work is the implementation and evaluation of alternative optimization techniques. Those earmarked for evaluation include Tabu Search and Genetic Algorithms.

In conclusion, it should be noted that the simulated annealing technique presented is not being proposed as a complete solution to the map generalization problem. Instead, the authors present it as a useful generalization tool to be used

Figure 10(b). Large scale OS data example. (b) Graphic conflict resolved by application of displacement, size enlargement, size reduction and deletion.

within some kind of broader generalization framework. For example, it could be used in a semi-automated setting by forming part of a set of generalization tools made available via a GIS toolbar. Alternatively, it could contribute to a more fully automated solution, maybe serving as a method available to certain object classes within an AGENT-like system, or maybe by acting as just one step in a pre-defined sequence of generalization operations (e.g. something akin to Bundy's 'internal agenda' (Bundy 1996) or the 'Global Master Plan' of Ruas and Plazanet (1996)).

Acknowledgments

Nathan Thomas is funded by EPSRC CASE Studentship 00802722, which is carried out in collaboration with the Ordnance Survey. The authors express thanks

to the Institut Géographique National and the Ordnance Survey for permission to use their data in parts of the work presented.

References

BUNDY, G. L., 1996, Automated cartographic generalization with a triangulated spatial model, PhD Thesis (available from The British Library).

BURGHARDT, D., and MEIER, S., 1997, Cartographic displacement using the snakes concept. In *Smati '97: Semantic Modelling for the Acquisition of Topographic Information from Images and Maps*, edited by W. Forstner and L. Plumer (Basel: Birkhauser) pp. 114–120.

CHRISTENSEN, J., MARKS, J., and SHIEBER, S., 1995, An empirical study of algorithms for point-feature label placement. *ACM Transactions on Graphics*, **14**, 203–232.

EMDEN-WEINERT, T., and PROKSCH, M., 1999, Best practice simulated annealing for the airline crew scheduling problem. *Journal of Heuristics*, **5**, 403–418.

HAIRE, K. R., and HARDY, P. G., 2001, Active agent based approaches to automated generalization. *Proceedings of 9th Annual GISRUK Conference*, 319–320.

HARRIE, L. E., 1999, The constraint method for solving spatial conflicts in cartographic generalization. *Cartography and Geographic Information Science*, **26**, 55–69.

HARRIE, L., and SARJAKOSKI, T., 2002, Simultaneous graphic generalization of vector data sets. *GeoInformatica*, **6**, 233–261.

HØJOLT, P., 2000, Solving space conflicts in map generalization: Using a finite element method. *Cartography and GIS*, **27**, 65–73.

JONES, C. B., BUNDY, G. L., and WARE, J. M., 1995, Map generalization with a triangulated data structure. *Cartography and Geographic Information Systems*, **22**, 317–331.

JONES, C. B., and WARE, J. M., 1998, Proximity search with a triangulated spatial model. *The Computer Journal*, **41**, 71–83.

KIRKPATRICK, S., GELATH, C. D., and VECCHI, M. P., 1983, Optimization by simulated annealing. *Science*, **220**, 671–680.

LAMY, S., RUAS, A., DEMAZEAU, Y., JACKSON, M., MACKANESS, W., and WEIBEL, R., 1999, The application of agents in automated map generalization. *Proceedings of 19th International Cartographic Conference*, 1225–1234.

LONERGAN, M. E., and JONES, C. B., 2001, An iterative displacement method for conflict resolution in map generalization. *Algorithmica*, **30**, 287–301.

MACKANESS, W. A., 1994, An algorithm for conflict identification and feature displacement in automated map generalization. *Cartography and Geographic Information Systems*, **21**, 219–232.

NICKERSON, B. G., 1988, Automated cartographic generalization for linear features. *Cartographica*, **25**, 15–66.

RUAS, A., 1998, A method for building displacement in automated map generalization. *International Journal of Geographical Information Science*, **12**, 789–803.

RUAS, A., and PLAZANET, C., 1996, Strategies for automated generalization. *Proceedings of 7th International Symposium on Spatial Data Handling*, **1**, 6.1–6.18.

RUSSELL, S., and NORVIG, P., 1995, *Artificial Intelligence: A Modern Approach* (Englewood Cliffs, New Jersey: Prentice-Hall), 1995.

STRIJK, T., and VAN KREVELD, M., 2002, Practical extensions of point labeling in the slider model. *GeoInformatica*, **6**, 181–197.

VARANELLI, J., and COHOON, J. P., 1995, A two-stage simulated annealing methodology. *Proceedings of 5th Great Lakes Symposium on VLSI, Buffalo NY*, 50–53.

WARE, J. M., and JONES, C. B., 1998, Conflict reduction in map generalization using iterative improvement. *Geoinformatica*, **2**, 383–407.

ZORASTER, S., 1997, Practical results using simulated annealing for point feature label placement. *Cartography and Geographical Information Systems*, **24**, 228–238.

Resolving Graphic Conflict in Scale Reduced Maps: Refining the Simulated Annealing Technique

Nathan Thomas[1], J. Mark Ware[1] and Christopher B. Jones[2]
[1]School of Computing, University of Glamorgan, Pontypridd CF37 1DL, UK
[2]Department of Computer Science, Cardiff University, Cardiff CF24 3XF, UK
jmware@glam.ac.uk

Introduction

Previous work has shown the potential for iterative improvement techniques to be used as part of an automated map generalisation solution (Ware and Jones, 1998; Ware et al, 2001). In particular, they present a simulated annealing algorithm that controls operations of displacement, deletion, reduction and enlargement of multiple map objects in order to resolve graphic conflict arising as a consequence of scale reduction. The algorithm adopts a trial position approach in which each of n discrete polygonal objects is assigned k candidate trial positions that represent the original, displaced, deleted, reduced and enlarged states of the object. This gives rise to a possible k^n distinct map configurations; the expectation is that some of these configurations will contain reduced levels of conflict. Each configuration has an associated overall cost, which can be computed. This overall cost combines both conflict cost (i.e. the extent to which acceptable minimum clearances between map objects are violated) and modification cost (i.e. the extent to which the map has been altered). Finding the configuration with least overall cost by means of an exhaustive search is not practical for realistic values of n and k. However, it has been shown that near optimum solutions can be found by using simulated annealing to direct a search through a subset of the configurations; thus effective resolution of graphical conflict can be achieved.

High-Order Feature Alignment

A shortcoming of the existing algorithm is that conflict cost depends only on the extent to which minimum clearance constraints are violated, and modification cost is calculated simply as a function of the degree to which each generalisation operator is applied. This approach can be regarded as primitive in that more subjective elements of map quality are overlooked. The specific problem that we address here is that of maintaining building alignment. One way in which a map can suffer deterioration in quality during generalisation is if a meaningful grouping of objects (or *high-order feature*) looses its shape to such an extent that the group is no longer recognisable. For example, displacing a particular building object might result in a row of building objects, representing a street, becoming misaligned or fragmented, thus rendering the street unrecognisable (Figure 1). Experiments show this type of problem to occur in practice.

Two possible solutions are being considered currently, both of which are conceptually straightforward. The first involves assigning a relatively high cost to situations where misalignment occurs. The hope is that such a strategy would both discourage offending displacements from taking place in the first place, and also stimulate remedial courses of action if and when misalignments do appear (e.g. realign buildings by performing additional displacements). The second solution involves high level feature modification (e.g. if a particular object is displaced, and that object forms part of a high level feature, then all other objects belonging to the same high level feature undergo the same displacement). There are possibly some added advantages with this approach in that the search space is made smaller (meaning the algorithm will run faster), modification operators can be applied in a more consistent fashion, and there is the possibility that remote solutions will be found more easily. A disadvantage is that conflict between objects belonging to the same high level feature will not be resolved (although post-processing of the data using solution 1 could be used to overcome this).

High-order features, such as a row of buildings representing a street, are not explicitly defined within source datasets (as is the case with OS MasterMap data). The first problem to solve is therefore that of banding together low-order features, such as an individual building, into higher-order features, such as streets. Finding a solution to the automated grouping of objects is a non-trivial task; previous work of note in this area has been carried out by Regnauld (1996) and Christophe and Ruas (2002). We intend adopting an approach in which building objects are grouped on the basis of a range of variables, including: minimum separating distance between buildings, initial alignment of buildings, relative orientation of buildings, proximity to roads, shape and size of buildings, and building attribute information.

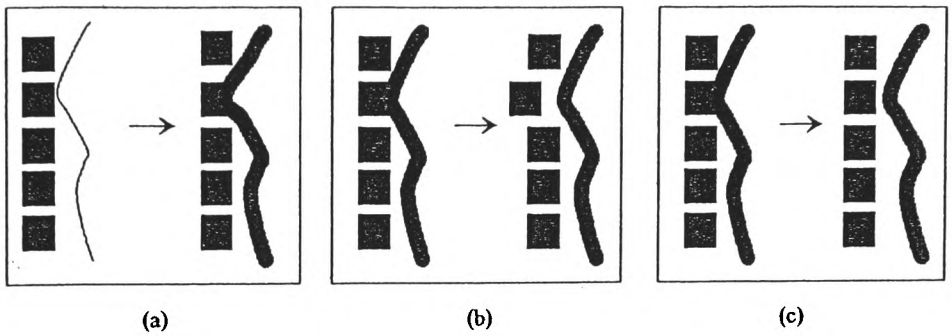


Figure 1. (a) Road symbolisation results in overlap with buildings. (b) Displacement of individual buildings resolves overlap but leads to misalignment. (c) Alignment maintained by treating row of buildings as a group.

Continuous Search Space

The use of fixed trial positions for the displacement of areal objects for the purpose of graphical conflict resolution has been shown to be successful. However in some circumstances, the use of a discrete search space can lead to problems. For example, consider Figure 2, in which the building feature is in conflict with a symbolised road feature.

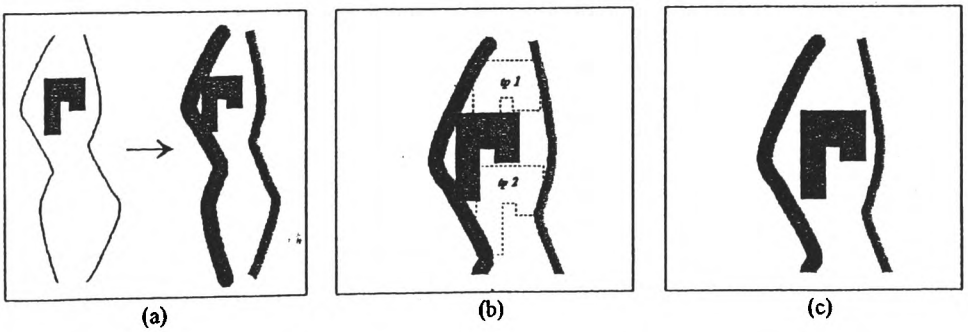


Figure 2. (a) Enough space exists for the object to move to in order to resolve spatial conflict. (b) However, none of the available trial positions resolve conflict. (c) The desired solution.

It can be seen that displacing the building slightly to the right would resolve all graphical conflict. Unfortunately, a trial position corresponding to such a displacement does not exist; in fact, displacing the building to any of its available trial positions results in further conflict. A similar, if perhaps less critical, problem is illustrated in Figure 3. As before, graphical conflict can be resolved by displacing the building feature slightly to the right, but, again, a trial position

corresponding to such a displacement does not exist. This time graphical conflict can be resolved by making use of one of the available trial positions; however, the problem we are faced with now is that the displacement appears excessive. The same problem can occur when applying the reduction operator.

A possible solution is to increase the resolution of the search space by adding to the number of displaced and reduced trial positions. This approach will be investigated, but there are two obvious limitations. First, it is clear that however many trial positions are added, the search space remains discrete and there is no guarantee that in any given situation the appropriate displacement or reduction will exist as a trial position. Second, increasing the number of trial positions increases the size of the search space and this is likely to have a negative impact on execution times.

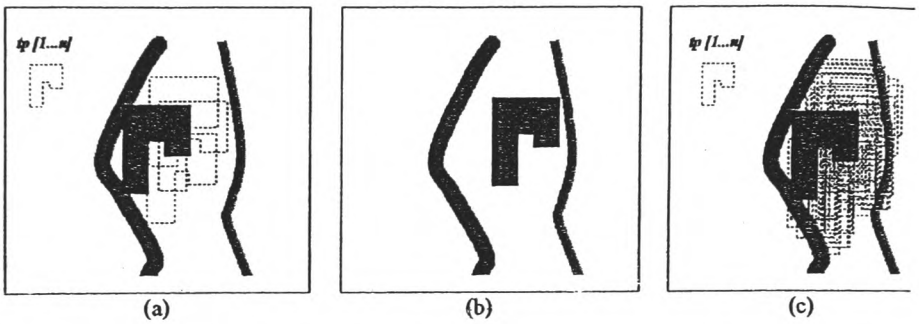


Figure 3. (a) Available trial positions. (b) Conflict is resolved, but displacement is excessive. (c) Increasing resolution of trial solutions might provide the required solution.

An alternative solution might be to adopt a continuous search space, as advocated by Strijk and van Kreveld (2002) for the purpose of point labelling. The idea we have is quite simple and involves the generation of random trial positions on the fly. The simulated annealing algorithm will work in very much the same way as before, with its main loop again beginning by choosing a modifiable object at random. However, instead of then choosing a trial position at random, the next step will involve the selection of an operator at random (i.e. displace, reduce, enlarge or delete) together with appropriate randomly generated operator parameters (i.e. if displace is selected then a random displacement vector is generated, and if reduce is selected then a random reduction factor is generated). The operator, and parameters, is then applied and the algorithm proceeds as before. By adopting this approach we will hopefully overcome the problem of having only a limited solution space. The hope is that the increased number of alternative realisations will produce improved results (i.e. minimum graphical conflict with least amount of modification).

Initial Results

Some of the ideas presented in previous sections have been implemented and tested using building polygon data extracted from OS MasterMap data and road centre lines taken from OS Oscar data. A simple building grouping function has been developed that groups building features together on the basis of both proximity and size. Each building feature is assigned to one group only. Output resulting from the application of this function is shown in Figure 4.

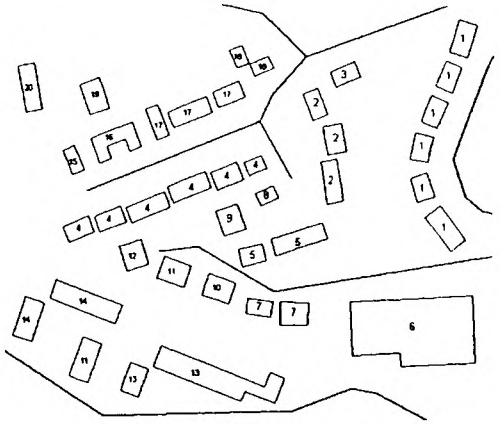


Figure 4. Building features, numbered according to group.

The simulated annealing algorithm described previously has been adapted so that map modifications are applied to groups as opposed to individual buildings. Consider a group g_1, consisting of three buildings b_1, b_2 and b_3. If g_1 is chosen to be modified (e.g. displaced) then the displacement is applied to b_1, b_2 and b_3. Figure 5 shows two examples of output produced with and without grouping.

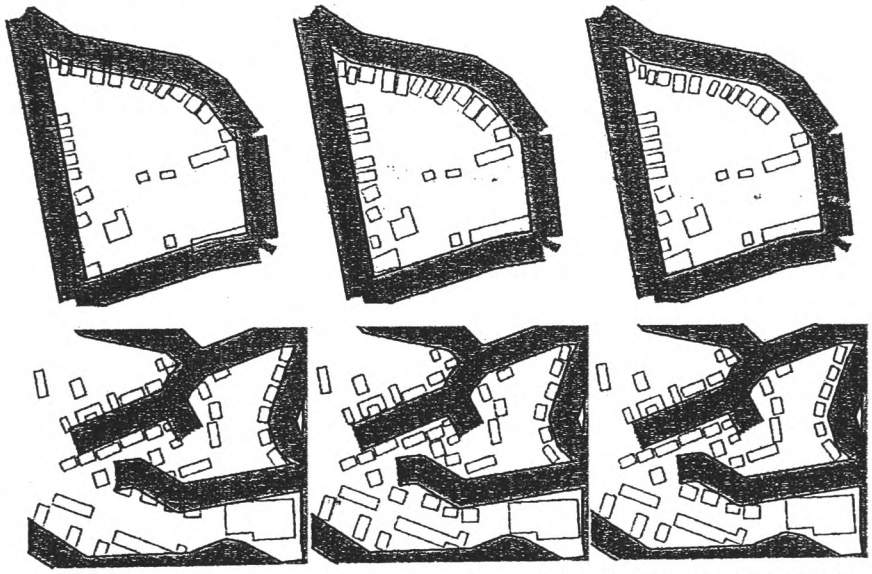


Figure 5. Symbolising road features results in graphic conflict (left). Application of simulated annealing (SA) to individual building features can lead to misalignment and fragmentation (middle). Application of SA to groups maintains high-order features (right).

The fixed trial position approach has also been replaced. Instead, modification operator parameters are generated randomly as part of the algorithm. This, in effect produces a continuous solution space. Figure 6 give example output produced using the alternative approaches.

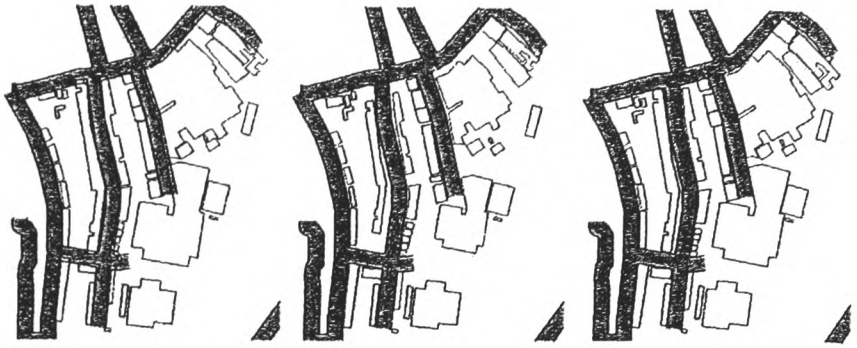


Figure 6. Symbolising road features results in graphic conflict (left). Application of SA using fixed trial positions results in excessive displacement and excessive reduction of building features (middle). Application of SA using randomly generated trial positions reduces displacement and reduction of building features (right).

Repeatable Results

Note that the simulated annealing implementation makes use of the *ran1* function (described in Press et al, 1992) to generate random numbers. The function is initialised with some arbitrary seed value. Each initialising value will typically produce a different random sequence, and solutions will vary. However, the same initialising value will always produce the same random sequence. This property provides a mechanism for reproducing previous solutions (which is achieved by simply keeping a record of seed values used).

Acknowledgement

This work is partially supported by EPSRC Case Award 00802722, in collaboration with the Ordnance Survey.

References

Christophe, S. and Ruas, A. (2002), Detecting building alignments for generalisation purposes, *Proceedings of 10th International Symposium on Spatial Data Handling*, pp.419-432

Regnault, N. (1996), Recognition of building clusters for generalisation, *Proceedings of 7th International Symposium on Spatial Data Handling*, pp.1-13

Ware, J.M. and Jones, C.B. (1998), Conflict Reduction in Map Generalisation Using Iterative Improvement, *Geoinformatica*, 2(4), pp.383-407

Ware, J.M., Jones, C.B. and Thomas, N. (2001), Map Generalization, Object Displacement and Simulated Annealing: Two Techniques for Execution Time Improvement, *Proceedings of GIS Research UK 2001 Conference (GISRUK'01)*, pp.36-38

Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P. (1992), Numerical Recipes in C (book), Cambridge University Press

Strijk, T. and van Kreveld, M. (2002), Practical Extensions of Point Labeling in the Slider Model, *GeoInformatica*, 6(2), pp.181-197

Biography

Nathan Thomas is a PhD student in the School of Computing at the University of Glamorgan. The title of his thesis is "Resolving Conflict in Scale Reduced Maps: A Simulated Annealing Approach". His project, due to finish at the end of 2003, is funded by an EPSRC Case Award, and is carried out in collaboration with the Ordnance Survey.

Maintaining Feature Alignment

J. Mark Ware, Nathan Thomas
School of Computing
University of Glamorgan
Pontypridd CF37 1DL, UK
jmware@glam.ac.uk, nthomas4@glam.ac.uk

Christopher B. Jones
Department of Computer Science
Cardiff University
Cardiff CF24 3XF, UK
c.b.jones@cs.cardiff.ac.uk

In previous work, the authors show the potential for iterative improvement techniques to be used as part of an automated map generalization solution ([1], [2]). In particular, they present a simulated annealing algorithm that controls operations of displacement, deletion, reduction and enlargement of multiple map objects in order to resolve graphic conflict arising as a consequence of scale reduction. The algorithm adopts a trial position approach in which each of n discrete polygonal objects is assigned k candidate trial positions that represent the original, displaced, deleted, reduced and enlarged states of the object. This gives rise to a possible k^n distinct map configurations; the expectation is that some of these configurations will contain reduced levels of conflict. Each configuration has an associated overall cost, which can be computed. This overall cost combines both conflict cost (i.e. the extent to which acceptable minimum clearances between map objects are violated) and modification cost (i.e. the extent to which the map has been altered). Finding the configuration with least overall cost by means of an exhaustive search is not practical for realistic values of n and k. However, it has been shown that near optimum solutions can be found by using simulated annealing to direct a search through a subset of the configurations; thus effective resolution of graphical conflict can be achieved (Figure 1).

A shortcoming of the existing algorithm is that conflict cost depends only on the extent to which minimum clearance constraints are violated, and modification cost is calculated simply as a function of the degree to which each generalization operator is applied. This approach can be regarded as primitive in that more subjective elements of map quality are overlooked. The specific problem addressed here is that of maintaining building alignment. One way in which a map can suffer deterioration in quality during generalization is if a meaningful grouping of objects (or *high level feature*) looses its shape to such an extent that the group is no longer recognizable. For example, displacing a particular building object might result in a row of building objects, representing a street, becoming misaligned, thus rendering the street unrecognizable (Figure 2). Experiments show this type of problem to occur in practice.

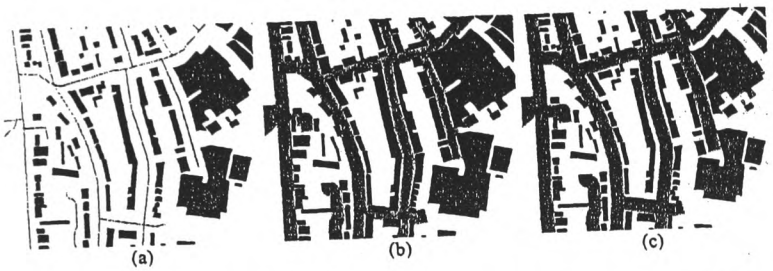


Figure 1. (a) Source road and building data. (b) Road symbolization causes graphical conflict. (c) Conflict resolved using simulated annealing algorithm by displacement, resize and deletion of buildings.

Two solutions are proposed, both of which are conceptually straightforward. The first involves assigning a relatively high cost to situations where misalignment occurs. The intention is that such a strategy will both discourage offending displacements from taking place in the first place, and also stimulate remedial courses of action if and when misalignments do appear (e.g. realign buildings by performing additional displacements). The second solution involves high level feature displacement (i.e. if a particular object is displaced, and that object forms part of a high level feature, then all other objects belonging to the same high level feature undergo the same displacement). There are added advantages with this approach in that the search space is made smaller (meaning the algorithm will run faster), modification operators can be applied in a more consistent fashion, and there is the possibility that remote solutions will be found more easily. A disadvantage is that conflict between objects belonging to the same high level feature will not be resolved (although post-processing of the data using solution 1 could be used to overcome this).

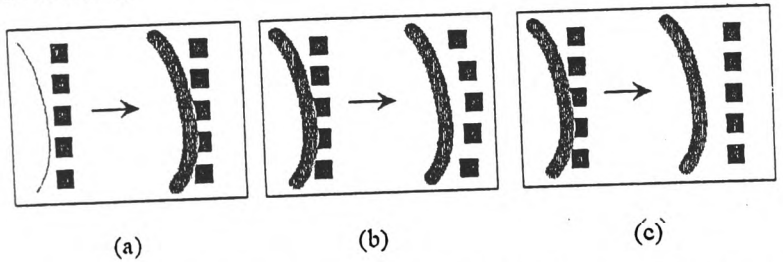


Figure 2. (a) Road symbolization results in overlap with buildings. (b) Displacement of individual buildings resolves overlap but leads to misalignment. (c) Alignment maintained by treating row of buildings as a group.

The conference talk will report on the initial stages of the implementation and evaluation of the proposed solutions. We will begin by describing the process by which streets are identified in the first instance (see [3] for previous work of note). In short, this is achieved by grouping building objects on the basis of a range of variables, including: minimum separating distance between buildings, initial alignment of buildings, relative orientation of buildings, shape and size of buildings, and building attribute information. Next, in relation to solution 1, we will describe a technique for measuring, and quantifying in map evaluation terms, the difference in alignment between an original grouping of buildings and any modified version that occurs during the running of the simulated annealing algorithm. This provides a mechanism for adding street misalignment costs to the overall cost associated with a particular map configuration. We will then go on to report the extent to which both solutions reduce the problem of misalignment due to displacement.

All experiments will make use of Ordnance Survey large scale data (i.e. Mastermap buildings and OSCAR road centre-line; in the case of Mastermap, the buildings are pre-processed using ArcGIS generalisation tools *AREAAGGREGATE* and *BUILDINGSIMPLIFY*). The simulated annealing algorithm, written in C, has been incorporated into an ArcView extension; it accepts shapefiles as input and produces shapefiles as output. The software will be demonstrated as part of the presentation.

Acknowledgments

This work is partially supported by EPSRC Case Award 00802722, in collaboration with the Ordnance Survey.

References

[1] Ware, J.M. and Jones, C.B., 1998, "Conflict Reduction in Map Generalisation Using Iterative Improvement", *GeoInformatica*, Volume 2, Number 4, pages 383-407.

[2] Ware, J.M., Jones, C.B. and Thomas, N., 2001, "Map Generalization, Object Displacement and Simulated Annealing: Two Techniques for Execution Time Improvement", *Proceedings of GIS Research UK 2001 Conference (GISRUK'01)*, pages 36-38.

[3] Regnauld, N., 1996, "Recognition of building clusters for generalization", *Proceedings of the 7th International Symposium on Spatial Data Handling* Volume 1, Delft, pages 4B.1-4B.14.

Map Generalization, Object Displacement and Simulated Annealing: Two Techniques for Execution Time Improvement

J.Mark Ware[1], Christopher B. Jones[2] & Nathan Thomas[1]
[1]GIS Research Centre, School of Computing, University of Glamorgan, Pontypridd, CF37 1DL, UK
[2]Department of Computer Science, Cardiff University, Cardiff, CF24 3XF, U.K
e-mail: jmware@glam.ac.uk

Introduction

In a previous work (Ware & Jones 1998, Ware et al 2000) the authors present an algorithm for reducing the spatial conflict that can arise as a consequence of displaying map data at scales smaller that its source scale. The algorithm addresses the specific problem of displacing multiple map objects in order to resolve graphic conflict. It adopts a *trial position* approach (similar to that used previously in point feature label placement e.g. Zoraster (1997)), in which each of n discrete polygonal objects is assigned k candidate trial positions into which it can possibly move. This results in a possible k^n distinct map configurations; the assumption is that some of these configurations will contain less conflict than the original. Finding the best (or, at least, an acceptable) alternative configuration by means of an exhaustive search is, however, not practical for realistic values of n and k (e.g. n=10 and k=8 gives rise to approximately 1,000,000,000 realizations). Instead the algorithm makes use of the *simulated annealing* search technique (Kirkpatrick, 1983), which has been shown to be successful in solving large optimization problems quickly e.g. (Emden-Weinert & Proksch, 1999). For our purposes, simulated annealing reduces the number of map realizations needing to be generated and evaluated.

Simulated Annealing Algorithm for Conflict Reduction

Our simulated annealing algorithm is an iterative procedure involving the repeated displacement of objects. Starting with an initial map configuration, an object and one of its trial positions are chosen at random; the object is then displaced to that trial position. If the displacement results in a map configuration with less conflict, then the object remains in the chosen trial position. If, however, the new map contains more (or the same amount of) conflict then the object is either returned to its previous position or remains in its new position, depending on some probability P. The process of attempting a random object displacement continues until some stop criterion is met (e.g. an acceptable solution is found or a pre-defined maximum amount of time has elapsed). At each iteration the probability P is dependant on two variables, ΔE (change in conflict) and T (current temperature). T is assigned a relatively high initial value; its value is decreased in stages throughout the running of the algorithm (until it reaches 0). At high temperatures poor displacements (large negative ΔE) will tend to be accepted, while at low temperatures poor displacements will tend to be rejected (although displacements resulting in small negative ΔE might still be accepted). When T=0 only good displacements (positive ΔE) are accepted. Note that the acceptance of

poor displacements is permitted so as to allow escape from locally optimal solutions. The initial value of T and the rate by which it decreases is governed by what is referred to as the *annealing schedule*. Generally, the higher the initial temperature and the slower the rate of change, the better the result (in conflict reduction terms); however, the processing overheads associated with the algorithm will increase as the rate of change in T becomes more gradual. In practice, a suitable annealing schedule is decided upon after some preliminary experimentation.

The experimental results reported by Ware & Jones (1998) show the simulated annealing approach to be successful in reducing the amount of spatial conflict in a map display reduced from 1:25,000 to 1:50,000 by up to 90%. It achieves this while at the same time limiting the number of realizations having to be visited (approximately 300,000 out of a possible 29^{321}). However, a shortcoming of the algorithm as it stands is that execution times remain too slow for it to be considered for use in applications requiring on-the-fly generalization (for example, it took approximately 40 seconds for the algorithm to generate and evaluate the 300,000 alternative map realizations referred to previously). In response to this shortcoming, this abstract suggests two ways in which the basic simulated annealing algorithm can be improved so as to reduce further the number of realizations having to be generated and evaluated.

Improvements to Algorithm

The first improvement is made by dividing the map into autonomous sub-regions (i.e. such that there is no possibility ever of objects in a particular region coming into conflict with objects in any other region). The reasoning here is that by dividing the data it is possible to produce a separate annealing schedule for each region, each schedule being set to meet the specific demands of its associated region. One problem to be solved here is finding a technique for dividing up the map. In some instances it may be possible to make use of naturally occurring regions, such as those formed by a road network or administrative boundaries. Other situations will require analysis of the distribution of objects in order to find groupings of objects that sustain no, or very little, influence from objects external to their group. Early experimental results, involving the use of a road network to divide the data, show that execution times can be reduced by up to 75% when adopting this approach.

A second improvement is achieved by adopting a two-stage approach. In the current algorithm the initial map configuration consists of each map object in its original or source position. Some authors suggest e.g. (Varanelli & Cohoon, 1993) the use of a two-stage approach. Here, to begin, a faster heuristic algorithm is used to replace the simulated annealing actions occurring at higher temperatures to produce the initial map configuration for a conventional simulated annealing algorithm. In tests carried out by the authors, the faster heuristic algorithm adopted is simulated annealing in conjunction with an annealing schedule that involves an initial high temperature followed by rapid cooling. The second stage again makes use of simulated annealing, this time with a much lower initial temperature followed by much slower cooling. Initial experimental results show that execution times can be reduced by up to 60% when adopting the two-stage approach.

Final Remarks

The full paper will describe the two improvements to the existing algorithm in greater detail. It will also give full experimental results, including those obtained when incorporating both improvements in a single algorithm. The paper will also give indication as to how the authors intend to develop the simulated annealing approach in the future; this will include ideas as to how additional generalization operators, such as object deletion and object amalgamation, might be integrated into the algorithm.

References

WARE, J.M. and JONES, C.B., 1998, Conflict Reduction in Map Generalization Using Iterative Improvement, Geoinformatica, 2(4): 383-407.

WARE, J.M., JONES, C.B. and LONERGAN, M.E., 2000, Map generalization by Iterative Improvement, First International Conference on Geographic Information Science (GIScience 2000).

ZORASTER, S., 1997, Practical results using simulated annealing for point feature label placement, Cartography and Geographical Information Systems, 24(4): 228-238.

KIRKPATRICK, S., GELATH, C.D. and VECCHI, M.P., 1983, "Optimization by simulated annealing", Science, 220:671-680.

EMDEN-WEINERT, T. and PROKSCH, M., 1999, "Best practice simulated annealing for the airline crew scheduling problem", Journal of Heuristics, 5(4): 403-418.

VARANELLI, J. and COHOON, J.P., 1993, "Two-Stage Simulated Annealing", ACM Physical Design Workshop, Lake Arrowhead, CA, April 1993.

Biographies

Mark Ware is a Senior Lecturer in the School of Computing at the University of Glamorgan. His research interests include terrain modelling, automated cartography (map generalization), automated environmental change detection, spatial indexing, multi-scale data structures and analysis and mapping of crime data.

Chris Jones is a Professor of GIS in the Department of Computer Science at Cardiff University. His research interests relate to problems of modelling, retrieval and presentation of spatial information in the geographical and geoscientific domains. A particular interest is the representation geographical information at multiple levels of detail and the associated problems of data integration, equivalence detection and map generalisation.

Nathan Thomas is a PhD student in the School of Computing at the University of Glamorgan. His project, which started in November 2000, is concerned with the development and evaluation of new procedures for automated map generalization. He is funded by an EPSRC CASE Studentship (00802722), which is carried out in collaboration with the Ordnance Survey.